Document N°: MANS900-81

Tutorial

First Edition

December 5, 2005

First Edition (November 2005)

Changes are periodically made to the information in this document. These changes will be incorporated into new editions of this document. MAX Technologies may make improvements and/or changes in the products and/or programs described in this document at any time.

Disclaimer

MAX Technologies Inc. makes no warranties as to the contents of this manual or the accompanying software. Although every effort has been made to ensure that the manual is accurate and that the software is reliable, MAX Technologies Inc. cannot be held responsible for any damages suffered from the use of this product.

How to reach MAX Technologies Inc. for Product Support

7005 Taschereau Blvd., Suite 350

Brossard, Quebec

Canada, J4Z 1A7

Tel.: (450) 443-3332

Fax: (450) 443-1618

Toll free: (800) 361-1629

www.maxt.com

Table of Contents

Chapter 1
Foreword

This tutorial is an introduction to using MAXIM. This tutorial is aimed at people wanting to familiarize themselves quickly with the capabilities of MAXIM. This is not a comprehensive reference manual. Many details are left out. It is recommended to read this document with the MAXIM user's manual close at hand.

This tutorial allows the user to build a project. The final project (named Tutorial.mpr) can be found in the MAXIM projects installation directory.

Chapter 2
Getting started

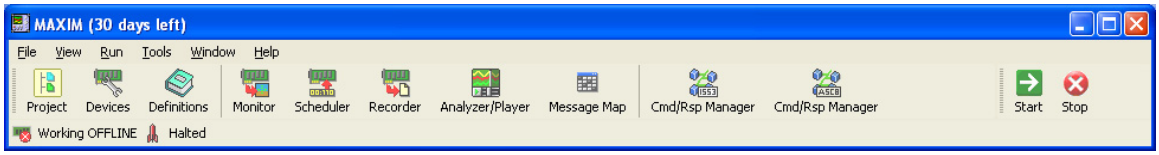
See MAXIM User's manual reference at:

Chapter 3: Getting Started

Starting MAXIM

A program icon named *MAXIM* is installed on the Windows desktop. Run it!

Once the loading completes, the MAXIM main window appears.



Chapter 3
Creating a new project

This chapter takes the user on a quick tour to set up a new project.

See MAXIM User's manual reference at:

Chapter 4: MAXIM's Main Window – File: New Project

Chapter 5: Project Manager – Project: Add to Project

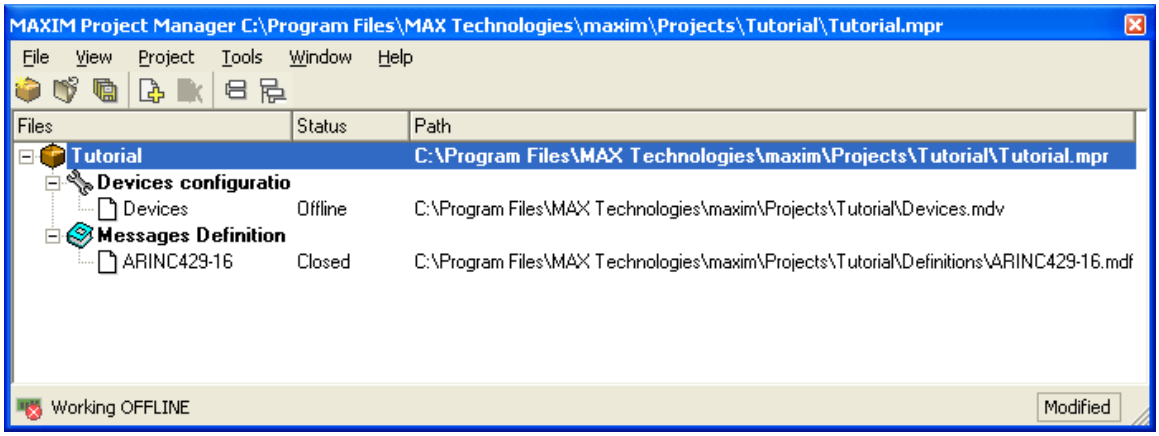
Chapter 6: Device Manager – Configuration

Chapter 4: MAXIM's Main Window – Tools: Preferences

Create a new project

Summary: To create a project, select New Project... from the Project menu.

To create a new project, select **File – New Project...** from the menu of the MAXIM main window. A file selection dialog opens to specify a name and location for the new project. Try it now! Choose any name for the project. After clicking **OK**, a project will be created.

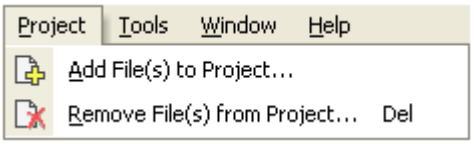


The newly created project contains a file named Devices that contains current devices configuration and the ARINC429-16 message definition file. This last file has been automatically added by using the "Default messages definitions files automatically added to each new project" setting of "Default project settings" in MAXIM preferences.

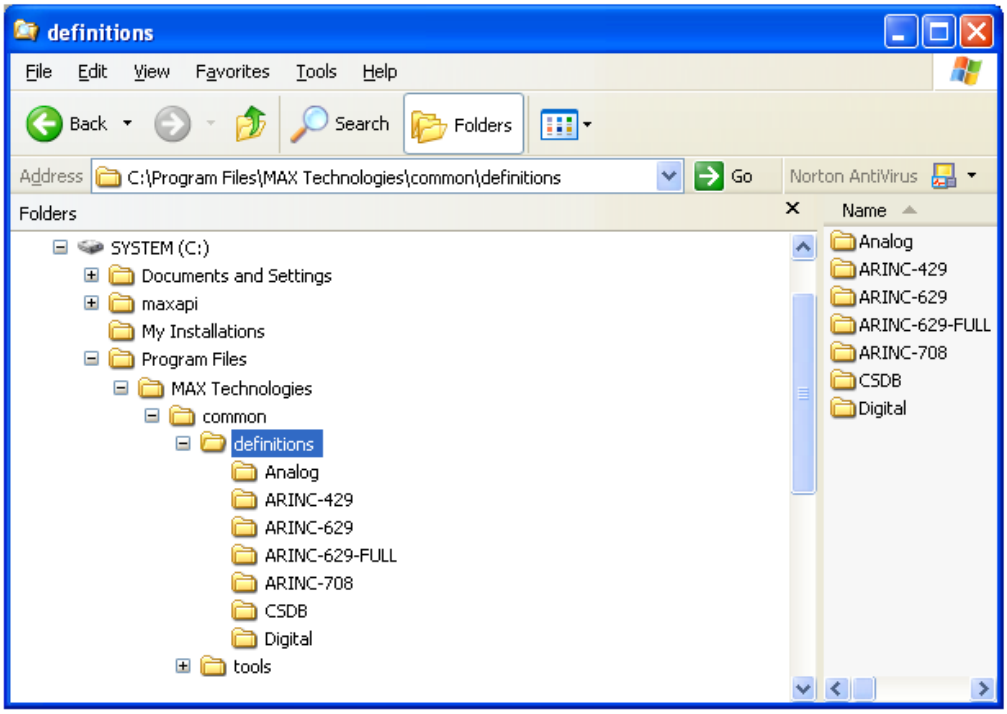
Add the message definition file(s)

Summary: To add message definition files, click the Add to project… button and select the files to add.

A message definition files can now be added. It will allow MAXIM to interpret data being transmitted or received. From the **Project** menu, choose **Add to project.** The **Open** dialog box opens.

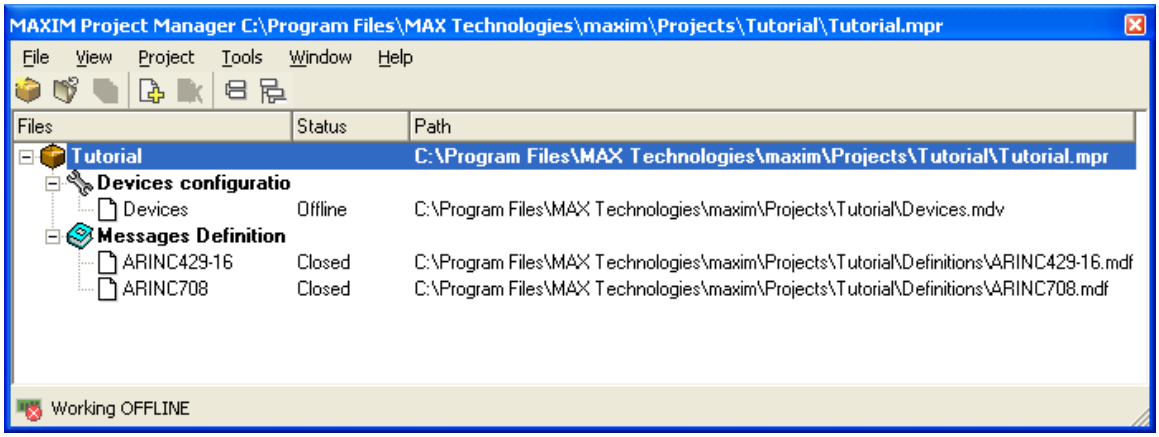


Go to the MAX Technologies definitions directory that is under the MAXIM installation directory:



For example, select the following message definition file:
ARINC-708.mdf

The project will look as follow:

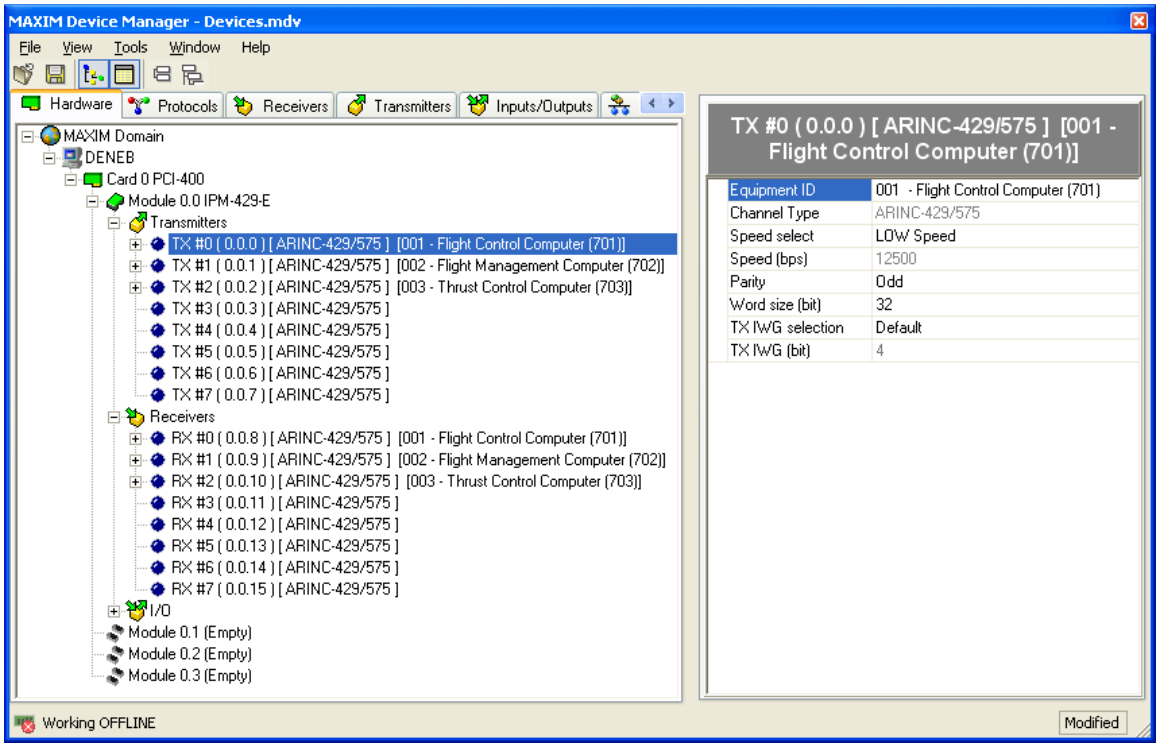


Configure hardware devices

Summary: To configure hardware devices, click the Devices… button from the MAXIM main window.

The Device manager allows the user to configure each channel of the system, assign equipment ID to channels, set bus speed, name the channels…

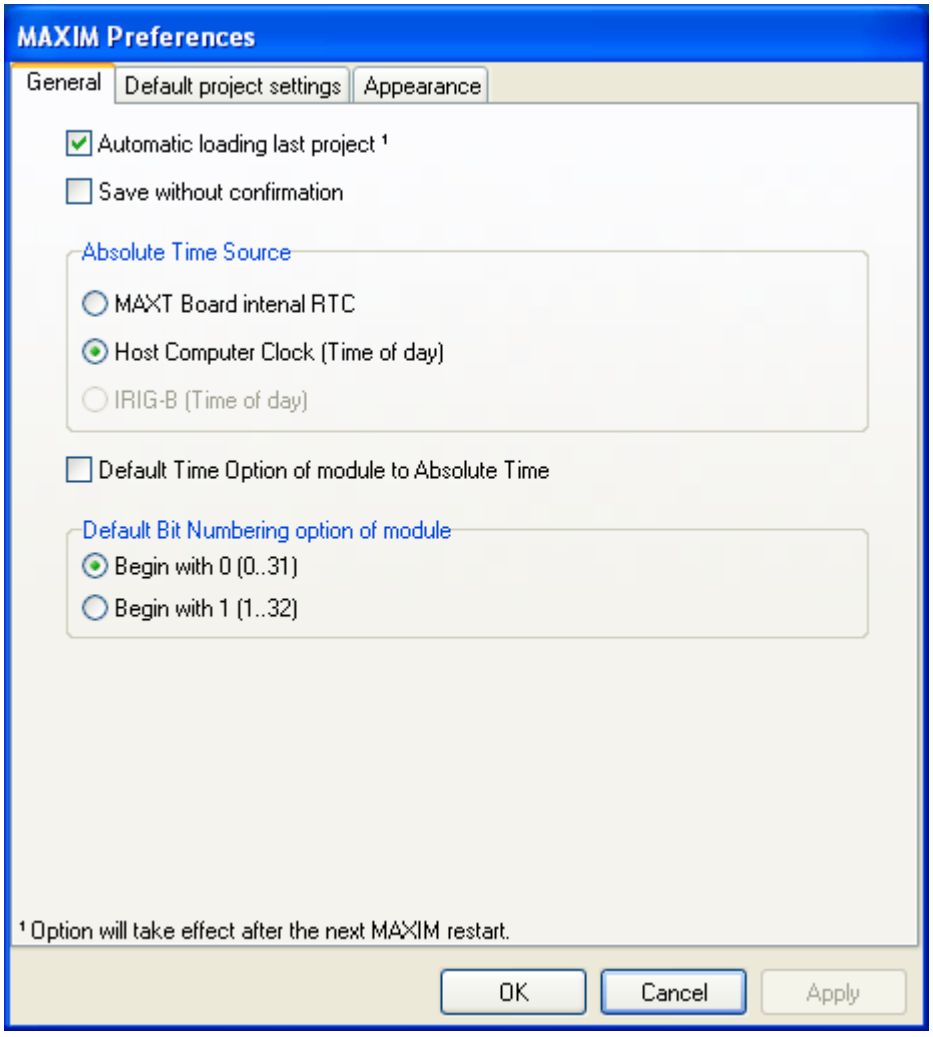
Click the **Save** button to record settings.



Configure general preferences

Summary: To configure a project, select Tools - Preferences... from the menu of the MAXIM main window.

To configure a project, select **Tools - Preferences...** from the MAXIM main window. The MAXIM preferences dialog opens to specify the MAXIM global options.



To use absolute time tags, choose the time source.
Note: The IRIG-B source is available if there is an IRIG-B time source in the system.

Chapter 4
Creating a Transmitting Session

This chapter takes the user on a quick tour to create a transmitting session.

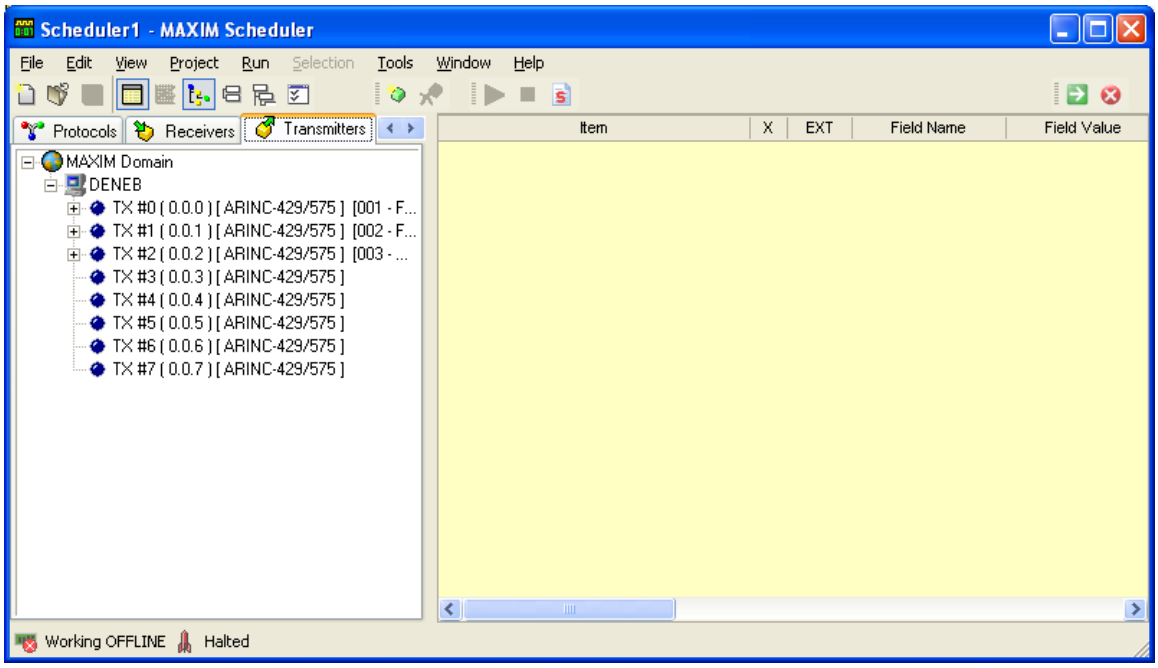
See MAXIM User's manual reference at:

Chapter 10: Scheduler

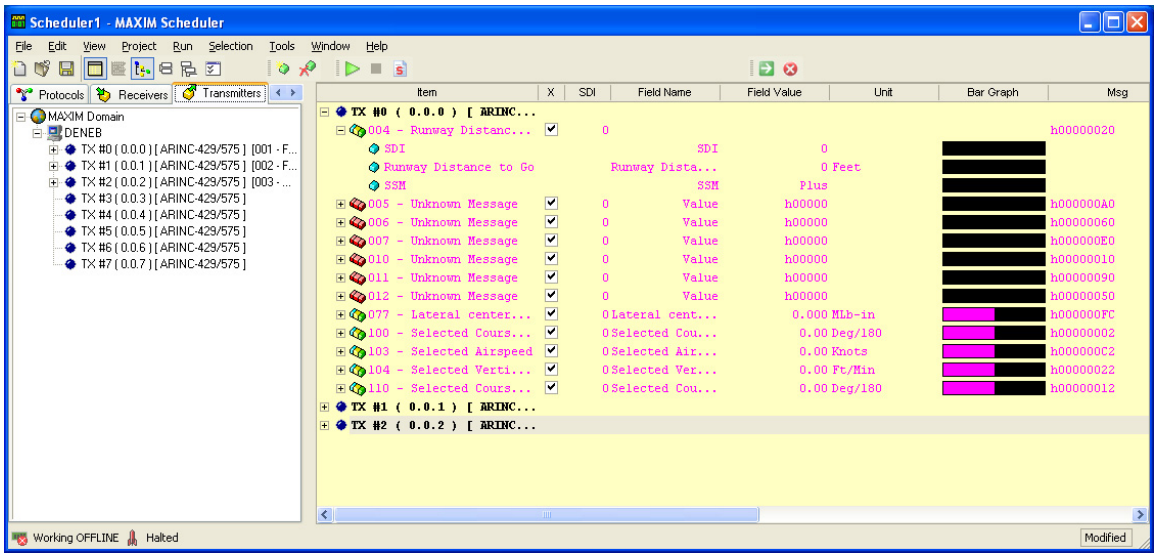
Create a new scheduler session

Summary: To create a scheduler, click the scheduler button from the MAXIM main window.

Click the scheduler button from the MAXIM main window to create a scheduler session.



Once the scheduler window appears, select and drag one channel from the left pane to the right pane of the window. When the channel is dragged, the **Add New messages** dialog box appears. Select some labels and click **OK**. Drag other channels and associate labels to them. Save the session by clicking the **Save** button.



Click the Start button to launch all schedules.

Chapter 5
Creating a Recording Session

This chapter takes the user on a quick tour to create a recording session.

See MAXIM User's manual reference at:

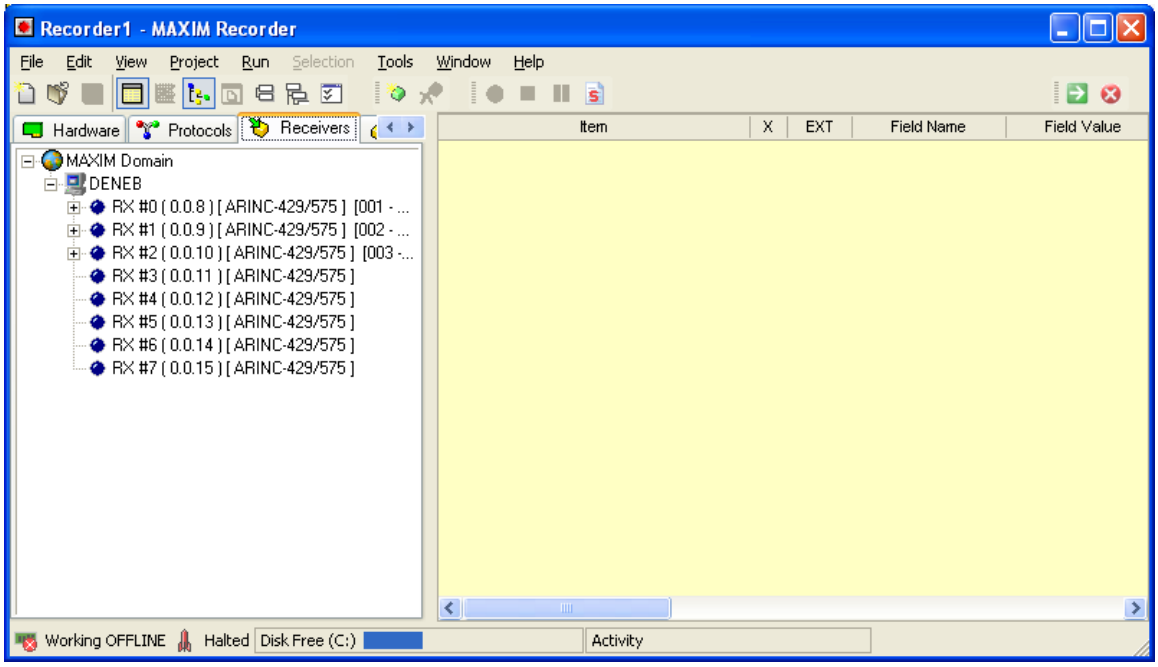
Chapter 8: Monitor/Recorder/Analyzer Common behavior

Chapter 11: Recorder

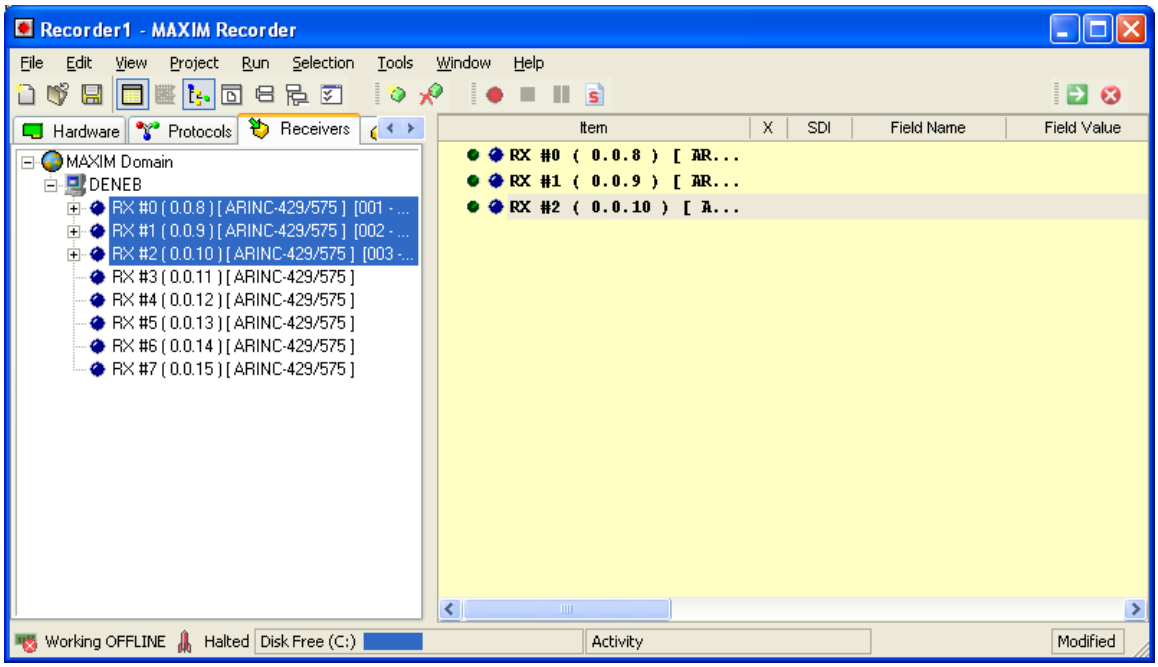
Create a new recorder session

Summary: To create a recorder, click the recorder button from the MAXIM main window.

Click the recorder button from the MAXIM main window to create a recorder session.

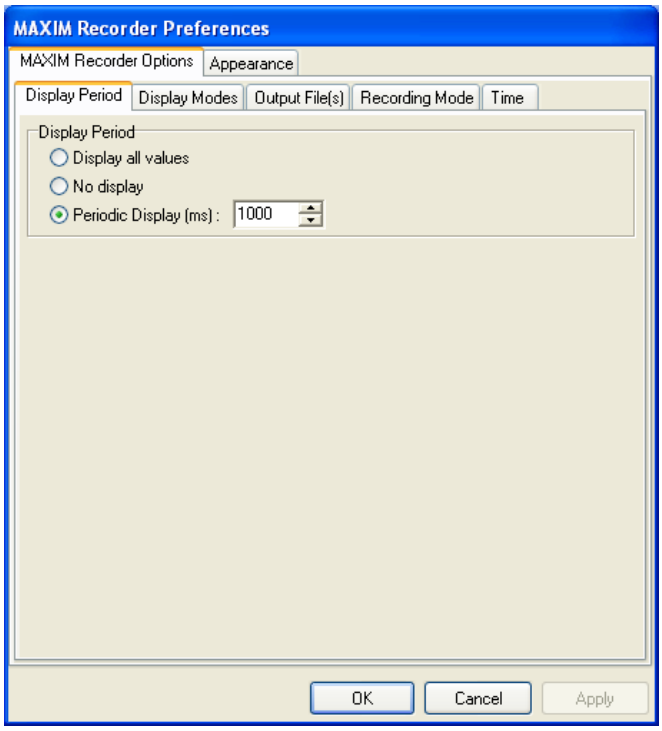


Once the recorder window appears, select and drag one channel from the left pane to the right pane of the window. When the channel is dragged, the **Add New messages** dialog box appears. Click on **close** to record all available labels on the bus. Save the session by clicking the **Save** button.


Recorder1 - MAXIM Recorder
File
Edit
View
Project
Run
Selection
Tools
Window
Help
Hardware
Protocols
Receivers
MAXIM Domain
DENEB
RX #0 (0.0.8) [ARINC-429/575] [001 - ...
RX #1 (0.0.9) [ARINC-429/575] [002 - ...
RX #2 (0.0.10) [ARINC-429/575] [003 -...
RX #3 (0.0.11) [ARINC-429/575]
RX #4 (0.0.12) [ARINC-429/575]
RX #5 (0.0.13) [ARINC-429/575]
RX #6 (0.0.14) [ARINC-429/575]
RX #7 (0.0.15) [ARINC-429/575]
Item
X
SDI
Field Name
Field Value
RX #0 (0.0.8) [AR...
RX #1 (0.0.9) [AR...
RX #2 (0.0.10) [A...
Working OFFLINE
Halted
Disk Free (C:)
Activity
Modified

Configure the recording session

Before starting the recorder, configure the recording session by clicking the **Preferences…** button located in the **Tools** menu.

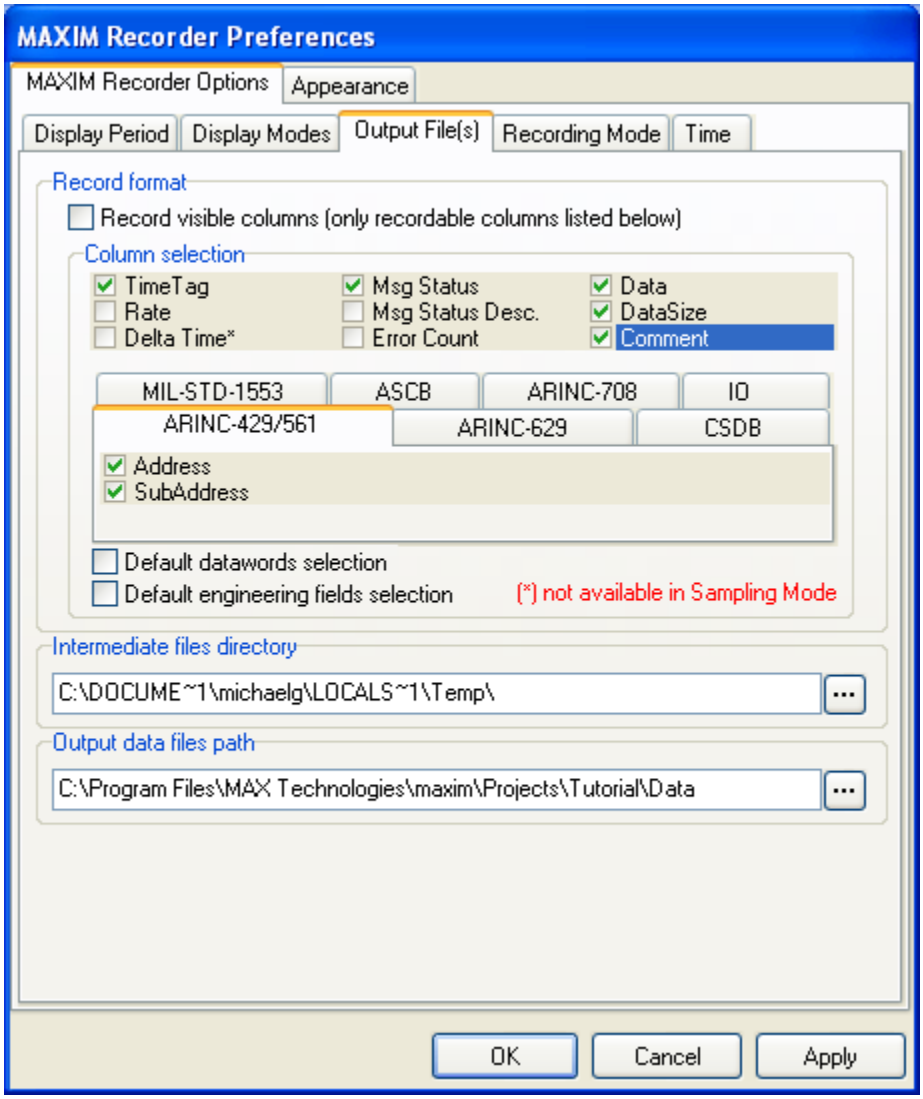


Preferences are grouped under two main tabs:

- The MAXIM recording options tab is used to configure the session
- The Appearance tab is used to configure the appearance of the recorder window.

For more details about the recorder preferences, refer to the MAXIM user's manual Monitor/Recorder/Analyzer Common behavior chapter – Tools: Preferences.

Let's configure the Output File(s) tab.



Select the record format by choosing columns to save in the recording file. To associate a comment to records, click the Comments check box.

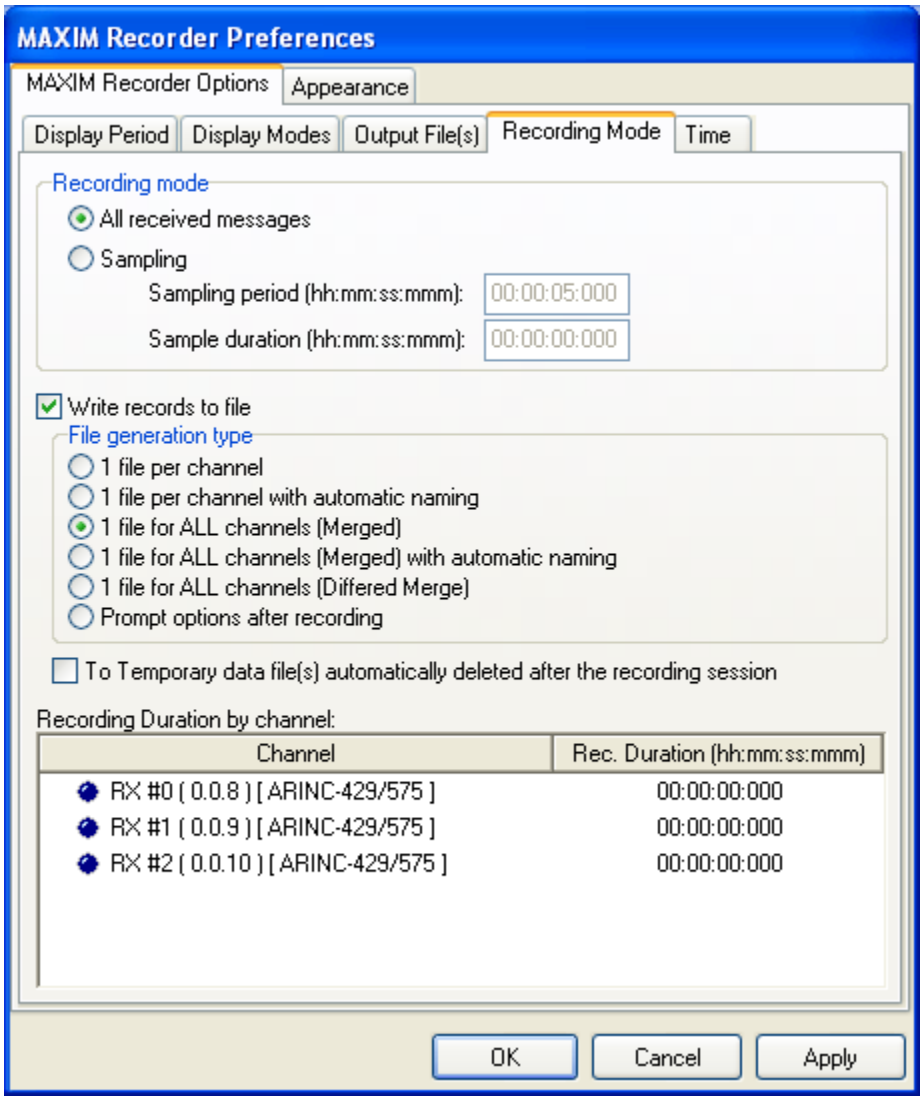
The two check boxes named "Default datawords selection" and "default engineering fields selection" indicate if fields are included in the file(s)for each received message.

It is very important to determine the information to save in the file(s). The performance of the acquisition system and the space that will be used on the disk depend heavily on these settings.

MAXIM can include engineering values in recorded files. However, to improve performance, the user can decide not to include them. The MAXIM Analyzer/Player module can be used to interpret acquired data after the recording session.

Unselect the two check boxes named "Default datawords selection" and "default engineering fields selection".

Let's configure the Recording Mode tab.



First, select the file generation type.

- **1 file per channel** means that for each channel, data will be stored in a separate file.
- **1 file per channel with automatic naming** means that for each channel, data will be stored in a separate file and MAXIM will automatically name files and copy them in a sub-directory of the data directory. Files are named MXDATA[CHN_SID].dat. If more than one file has the same name, an index is appended to the filename MXDATA[CHN_SID](x).dat. The directory is named RecorderName_RECORDED_FILES(x) where x is an index. The first time, the directory is named RecorderName_RECORDED_FILES with no index.
- **1 file for ALL channels (merged)** means that for all channels, data will be stored in a single file: Data will be sorted by the time tag in the file.
- **1 file for ALL channels (Merged) with automatic naming** means that for each channel, data will be stored in a unique file and MAXIM will automatically name the file and copy

it in the data directory. The file is name MXDATA[X.X.X](x).dat where x is an index. The first time, the file is simply name MXDATA[X.X.X].dat with no index. Data is sorted by the time tag in the file.

- **1 file for ALL channels (Differed Merge)** means that the merging process will be postponed to a later time, until the File Merger command is used. Data is sorted by the time tag in the file.

One file can be selected for all channels (Merged).

The recording mode radio button group allows the user to configure how the data will be acquired before storing them in the file.

"All received messages" means that the recorder will catch all the data on the bus.

"Sampling" means that the recorder will periodically catch the data on the bus. The period is defined by the Sampling period edit box. The Sample duration can also be specified.

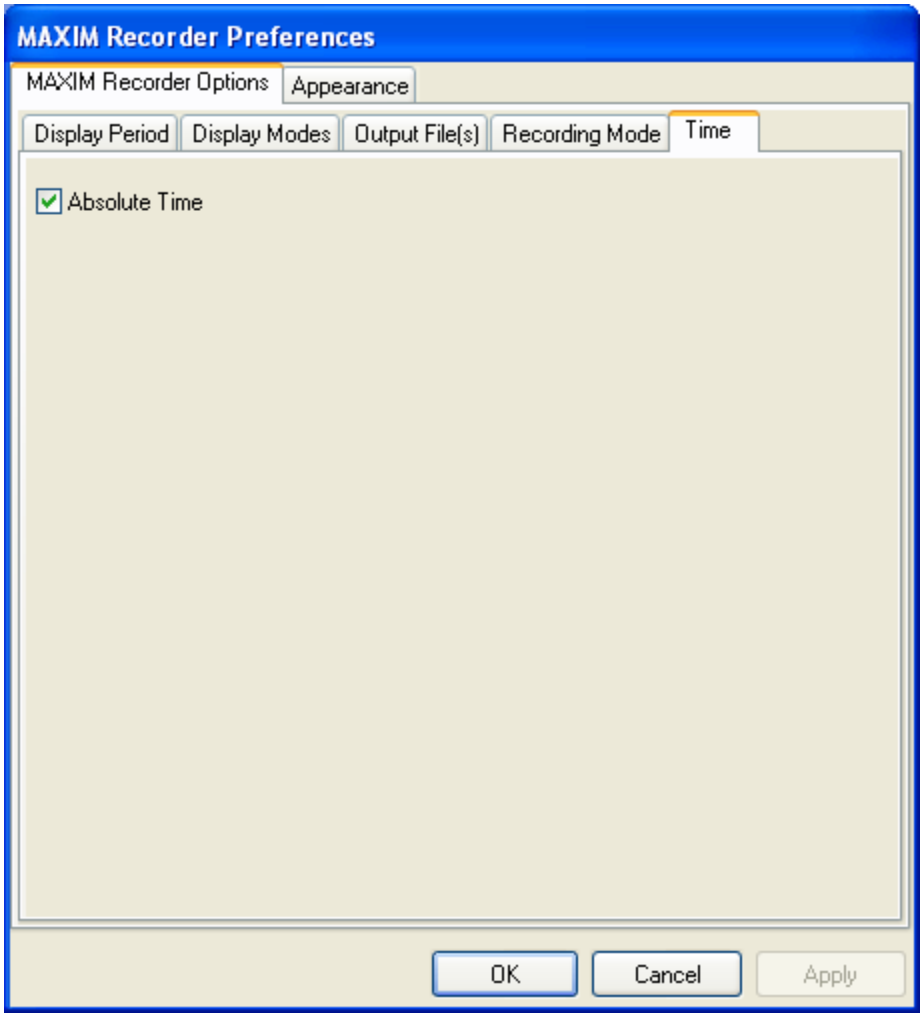
All received messages can be selected.

The user can select to write records to a file named by the user at the end of the acquiring session.

The recording duration channel list allows the user to enter an acquisition time for each channel of the recorder. If an acquisition time is entered for each channel, the recording session will automatically stop after the longest time entered that has elapsed.

The recording session is stopped manually in this example. Leave each acquisition time to 00:00:00:000.

Let's configure the Time tab.

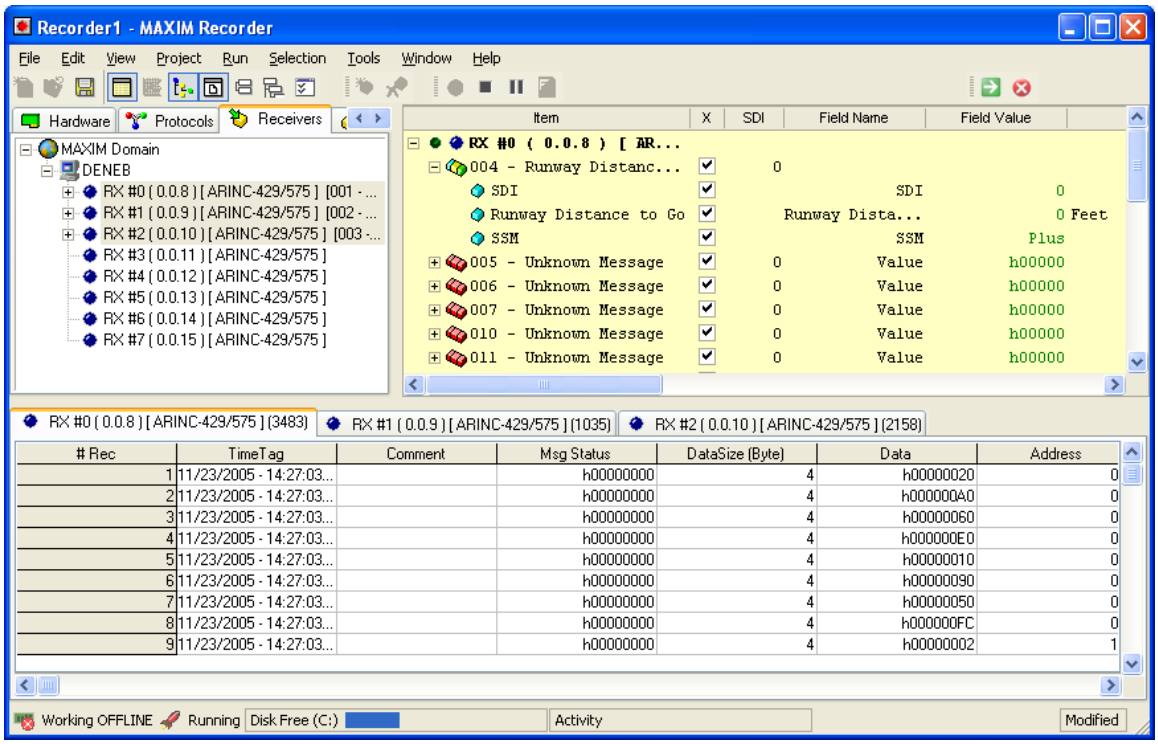


By default, the recording session stores time tags in relative mode. If the Absolute time check box is checked, each time tag will be stored as an absolute time. An absolute time tag is made up of the date and the time. The absolute time is stored as GMT time but can be displayed as a Local time at any time.

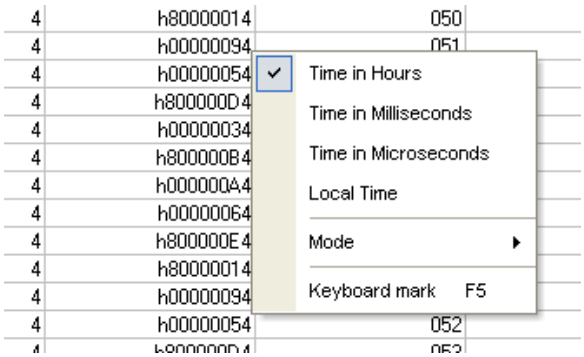
If the Absolute time check box is checked, the absolute time source must be configure in the MAXIM preferences.

The Absolute time box can be checked.

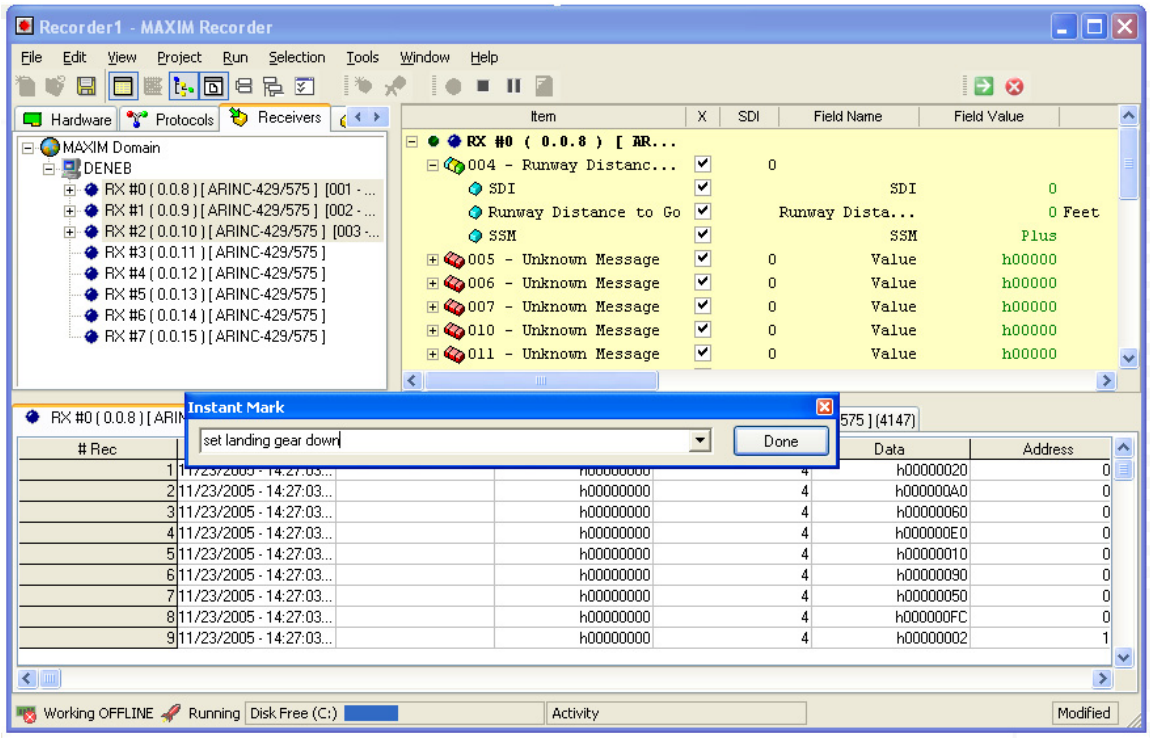
Click the Start button to start recording. If buses are active, the recorder window will look like this:



During the recording, records being acquired are displayed. Right-click on the file viewer to change the time format or the file viewer appearance.

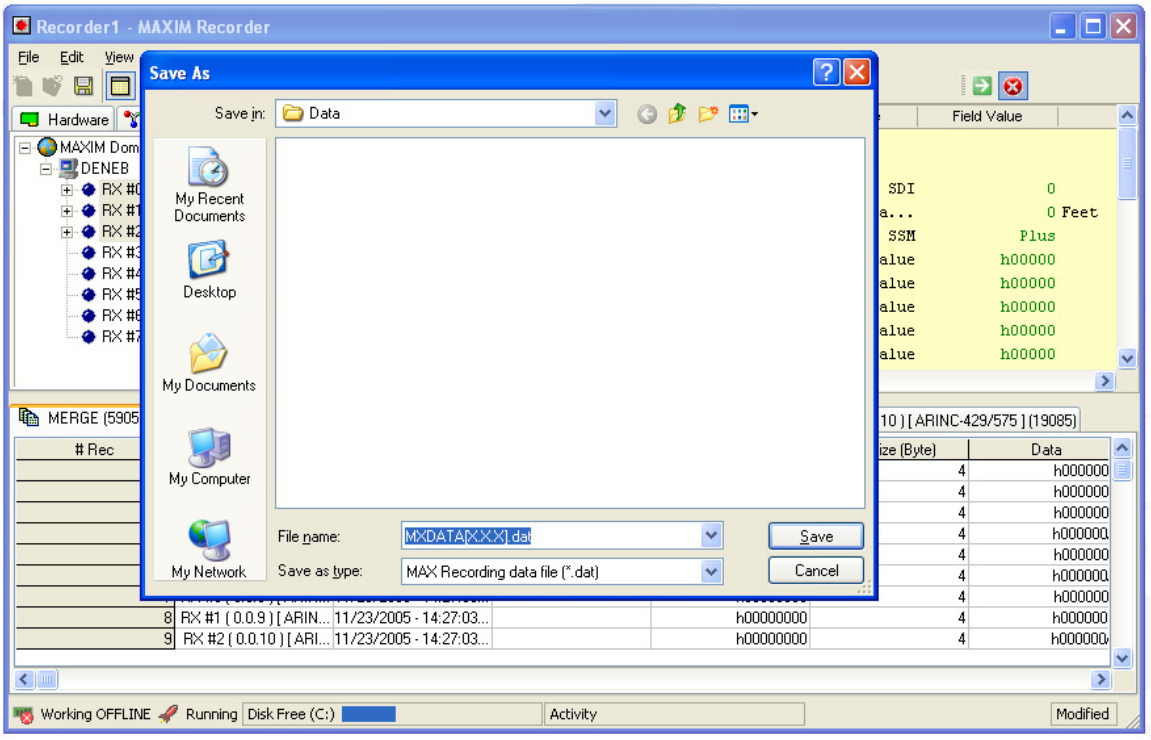


Click on the Keyboard mark menu to mark and comment a record.



Write a comment and/or type the “enter” key to mark and comment the current record.

Click the stop button to terminate the recording session.



Save the recording file.

Chapter 6
Analyzing a data file

This chapter takes the user on a quick tour to analyzing a data file.

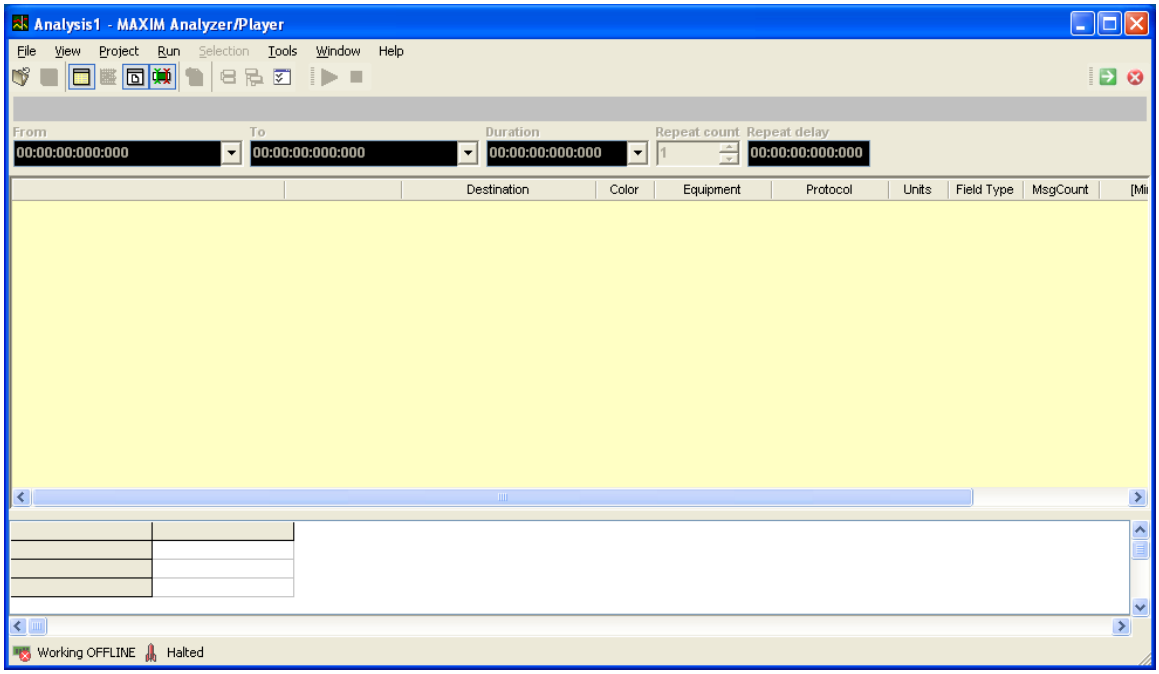
See MAXIM User's manual reference at:

Chapter 13: Analyzer/Player

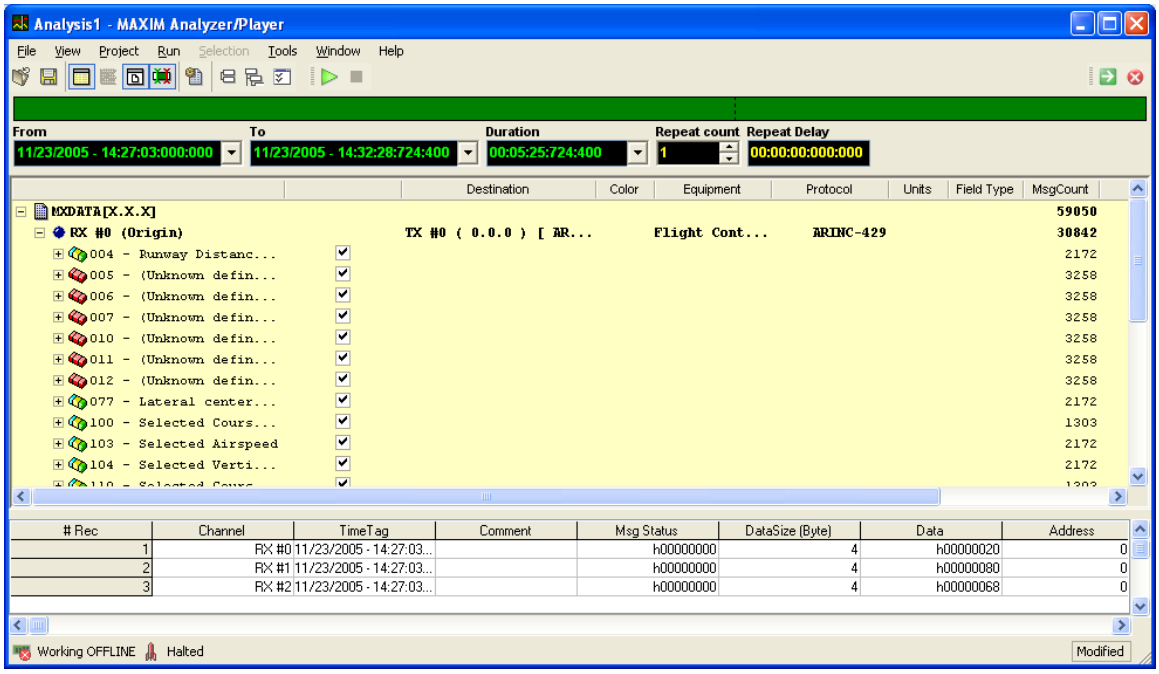
Create a new analyzer session

Summary: To create an analyzer, click the Analyzer/Player button from the MAXIM main window. Open the file to analyze.

Click the Analyzer/Player button from the MAXIM main window to create the Analyzer session.



Once the Analyzer/Player window appears, from the **File** menu, choose **Open.** The **Open** dialog box opens. Select the file to analyze.

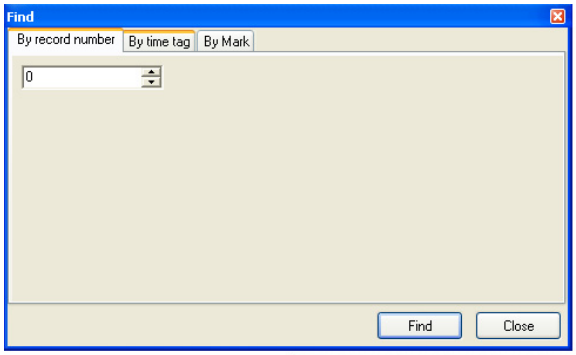


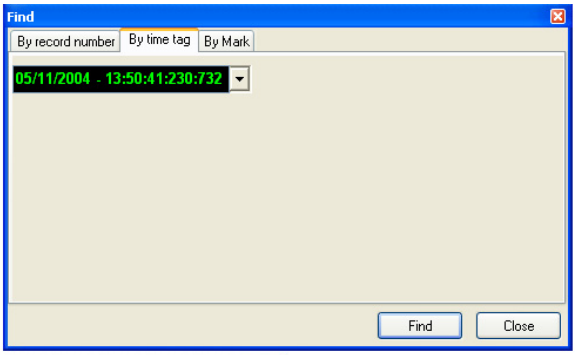
Save the Analysis session. From the **File** menu, choose **Save**. The **Save** dialog box opens.

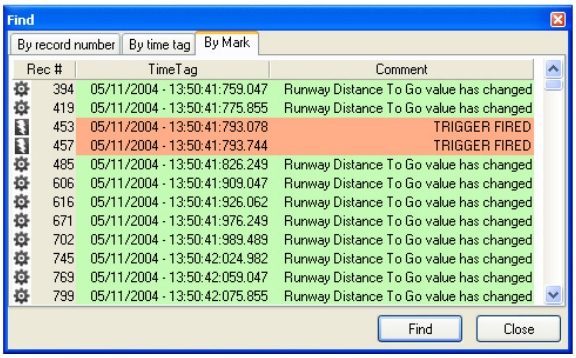
Finding a record

Summary: To find a record, select Find record... from the Tools menu.

To find a specific record, select **Tools** – **Find record...** from the menu. A find dialog opens to search a record using a record number, a timetag or a mark.



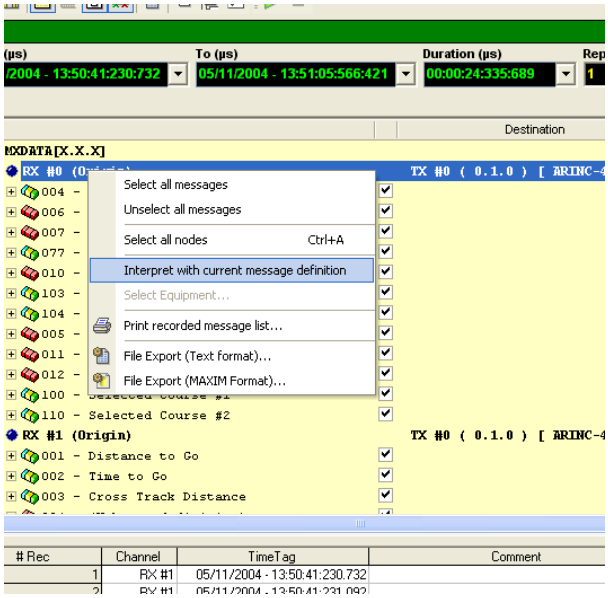




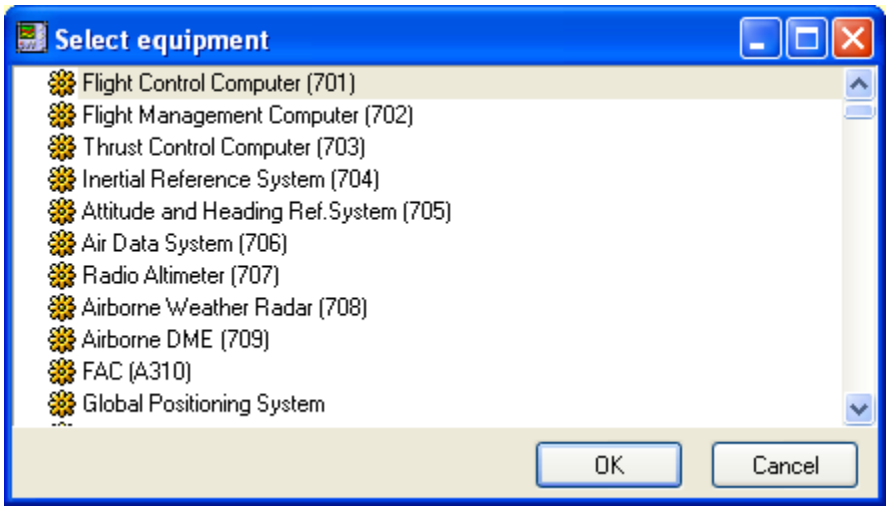
Post-interpret acquired data

The MAXIM Analyzer/Player module allows the user to interpret recording data according to the message definitions present in the current project.

Select a channel node and right-click to display the contextual menu.

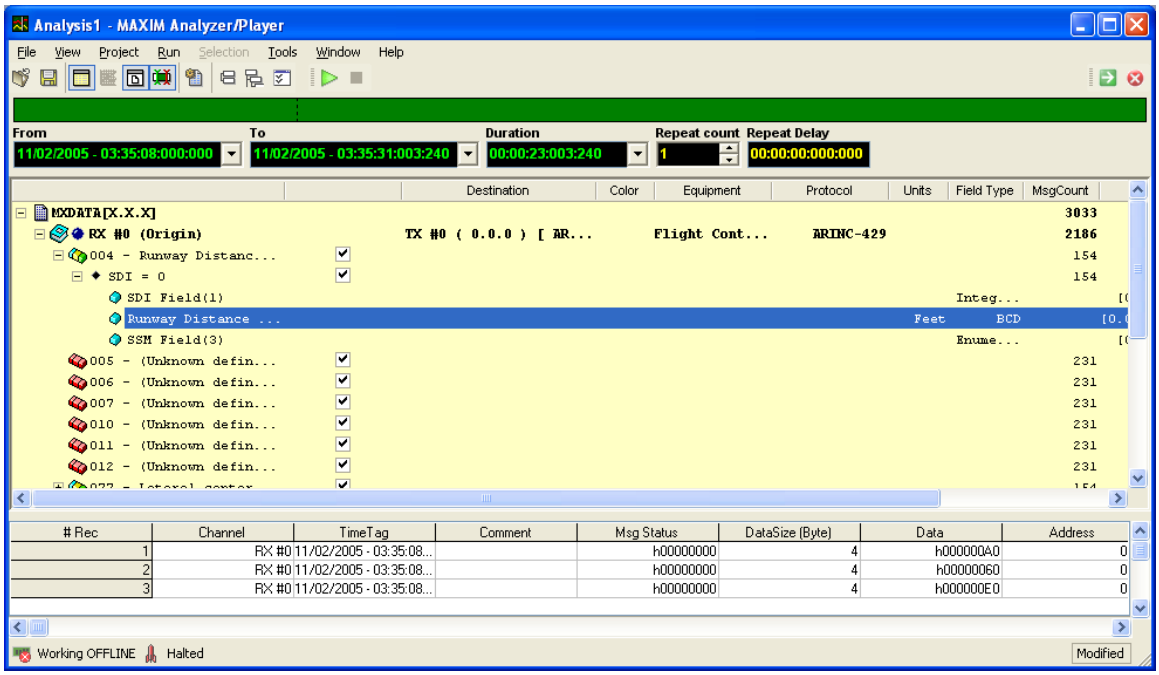


Click “Interpret with current message definition” menu. The Equipment Id selector window opens.



Let’s choose the “Flight Control computer (701) equipment”.

Each label that has a definition in the message definition file is now interpreted. The following figure illustrates the interpretation of the label 004.


Analysis1 - MAXIM Analyzer/Player
File View Project Run Selection Tools Window Help
From
11/02/2005 - 03:35:08:000:000
To
11/02/2005 - 03:35:31:003:240
Duration
00:00:23:003:240
Repeat count
1
Repeat Delay
00:00:00:000:000
Destination Color Equipment Protocol Units Field Type MsgCount
MXDATA[X.X.X] 3033
RX #0 (Origin) TX #0 (0.0.0) [AR... Flight Cont... ARINC-429 2186
004 - Runway Distanc... 154
SDI = 0 154
SDI Field(1) Integ...
Runway Distance ... Feet BCD
SSM Field(3) Enume...
005 - (Unknown defin... 231
006 - (Unknown defin... 231
007 - (Unknown defin... 231
010 - (Unknown defin... 231
011 - (Unknown defin... 231
012 - (Unknown defin... 231
Rec Channel TimeTag Comment Msg Status DataSize (Byte) Data Address
1 RX #0 11/02/2005 - 03:35:08... h00000000 4 h000000A0 0
2 RX #0 11/02/2005 - 03:35:08... h00000000 4 h00000060 0
3 RX #0 11/02/2005 - 03:35:08... h00000000 4 h000000E0 0
Working OFFLINE
Halted
Modified

Export acquired data

Export all or parts of the interpreted file. Click **File export (MAXIM format)** from the **Tools** menu to export the file using a MAXIM file format. Click **File export (Text format)** from the **Tools** menu to export the file using a CSV file format.

Chapter 7
Adding a script in a recording Session

This chapter takes the user on a quick tour of a recorder session script.

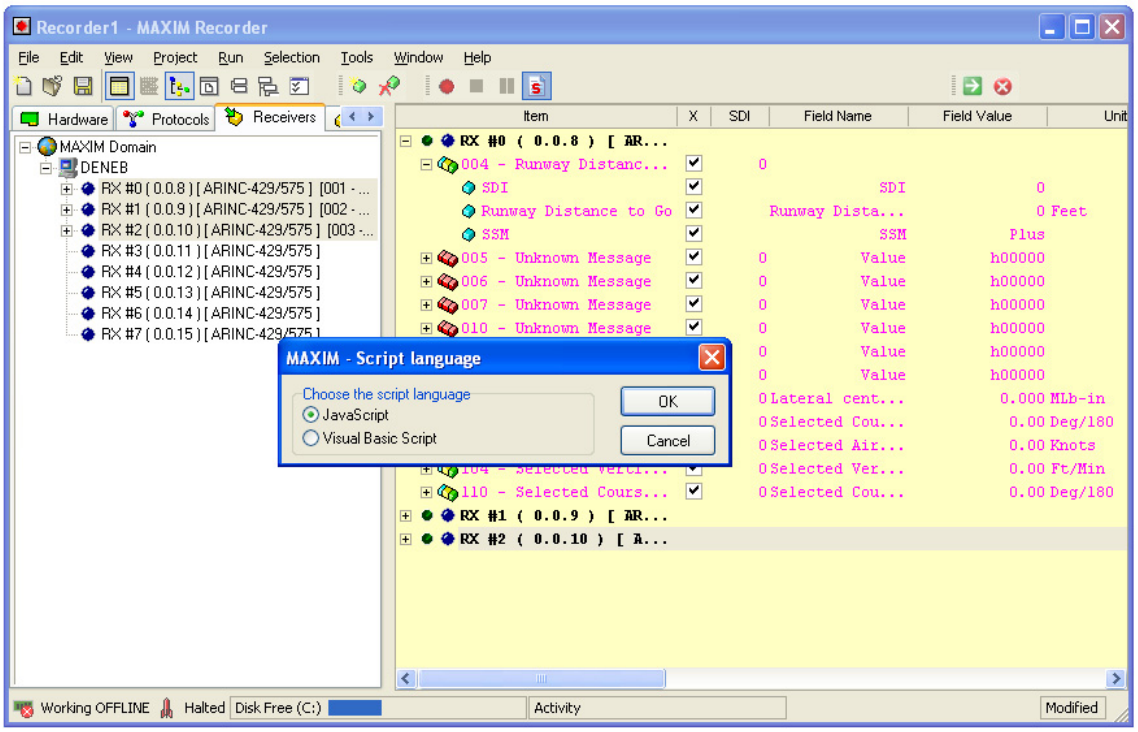
See MAXIM User's manual reference at:

Chapter 13: MAXIM Script Designer

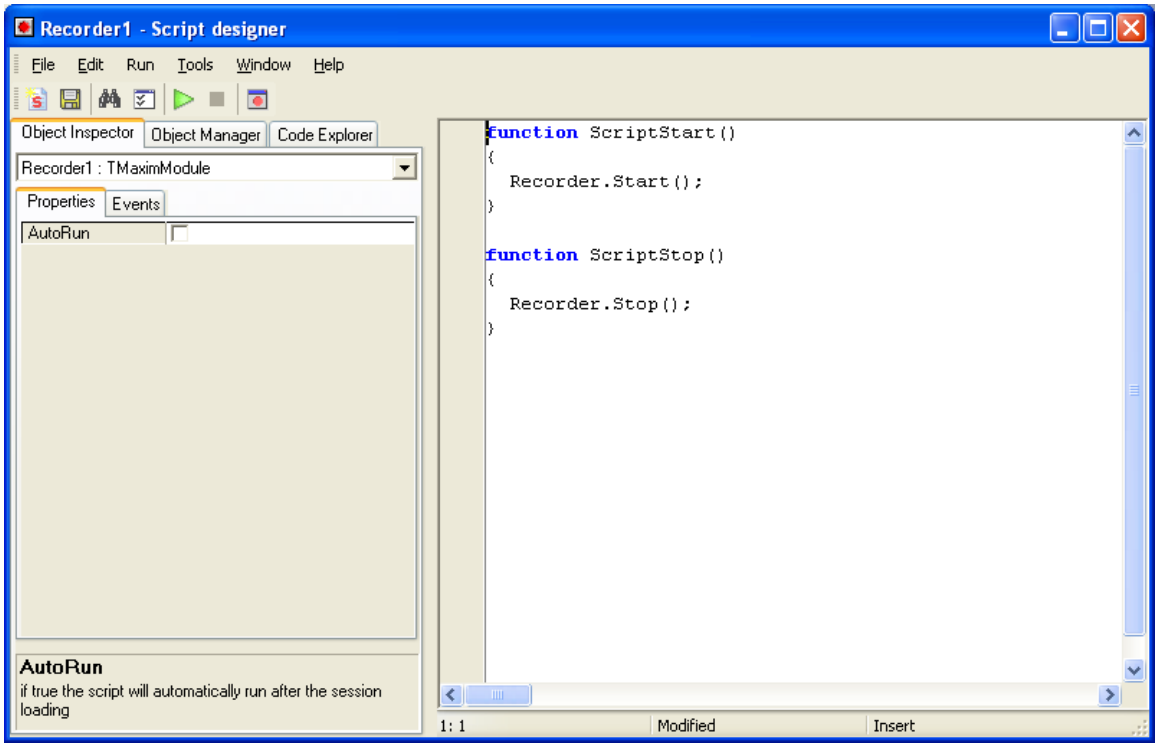
Create a script in an existing recorder session

Summary: To create a script, click the Script menu from the Tools menu.

Open an existing recorder session. Select **Script** from the **Tools** menu.



Choose for Example JavaScript.



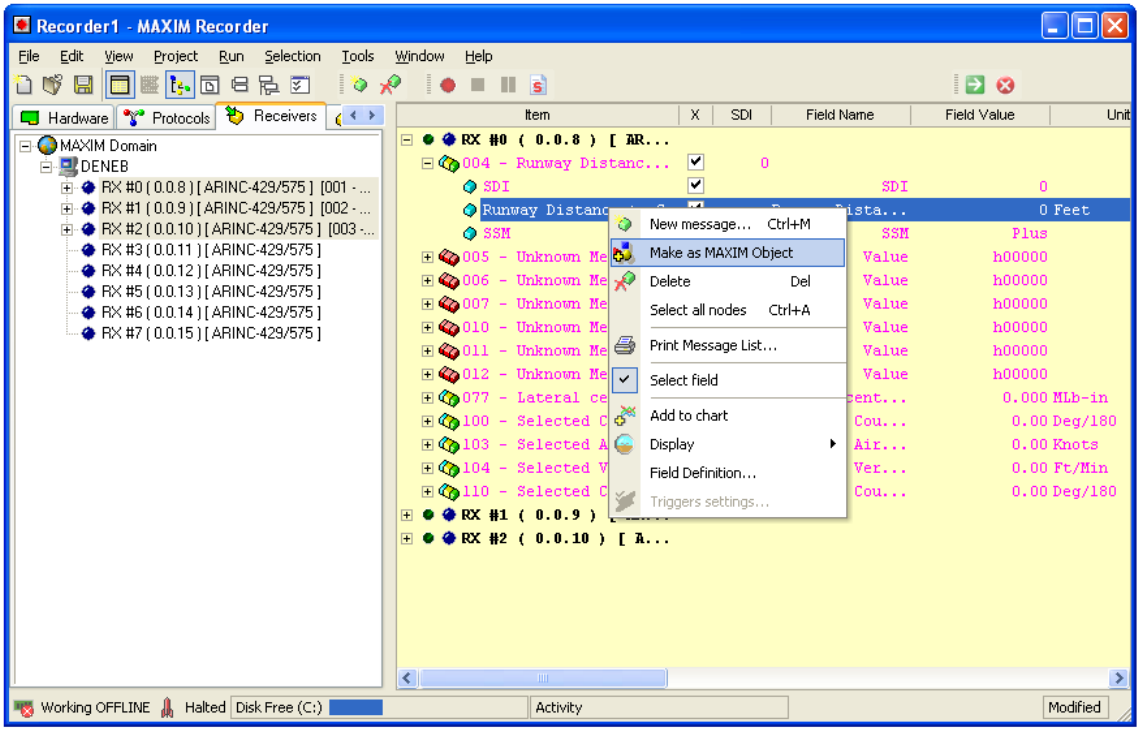
The left part of the window allows the user to select objects defined in the recorder session. The right part allows the user to edit the code.

Quit the script designing mode by choosing **Exit** from the **File** menu.

Manage a trigger with a script

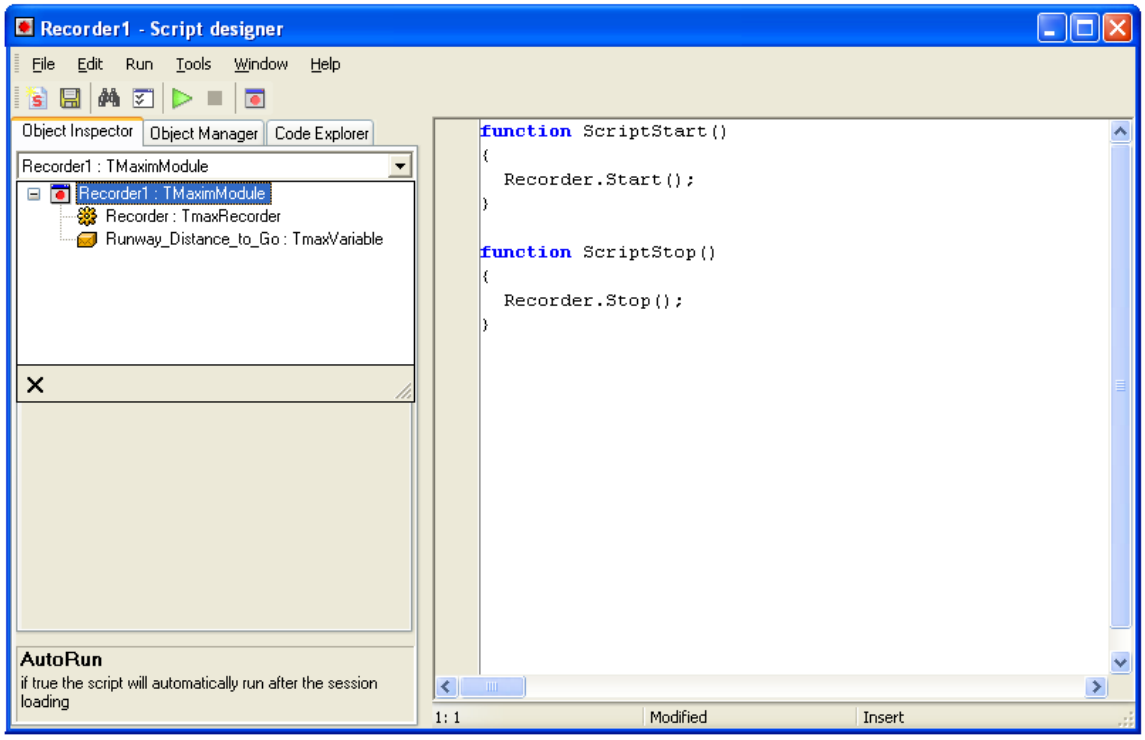
To illustrate MAXIM scripts, a script that checks an engineering value is created. It launches the recording session if the value exceeds a threshold.

In this example, select the Runway Distance to Go Field from the 004 label of the RX#0 Channel.



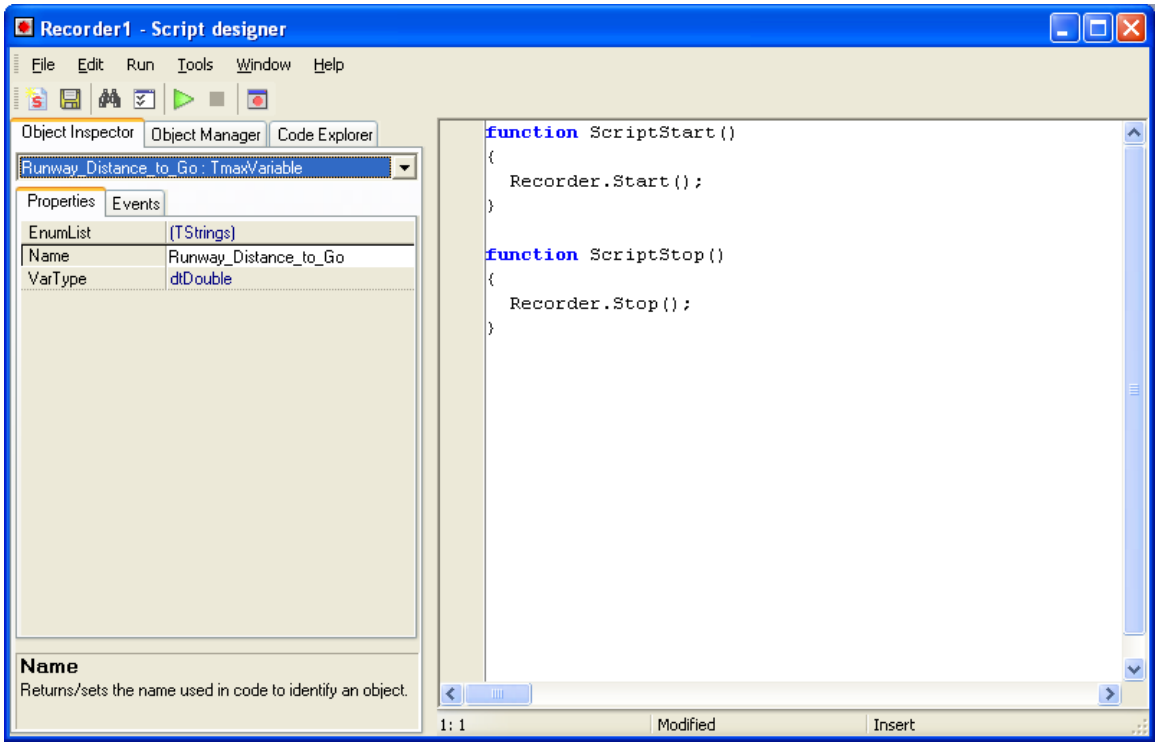
Right-click and select “Make As MAXIM Object”. The “Runway Distance to Go” field will now be accessible to the script.

Select **Script** from the **Tools** menu.

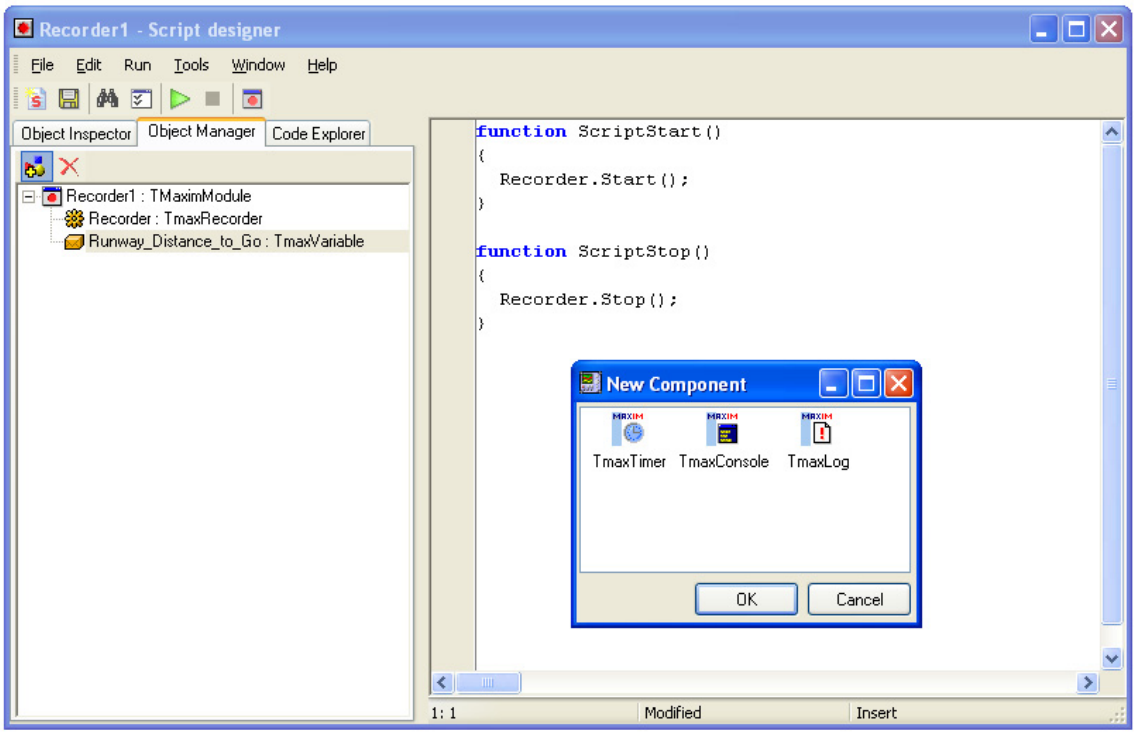


From the Object Inspector tab, select objects that are defined in the current session. There is a Recorder object that represents the recorder engine and the object Runway_Distance_to_Go that have just been created.

Select the “Runway_Distance_to_Go” object.



All objects exposed by MAXIM appear in the left section of the window, under the Object Manager tab. In this example, the Recorder Session object and one parameter are available. The user can use this tab to add or remove objects as shown here with the New Component window. A TmaxConsole will be added.



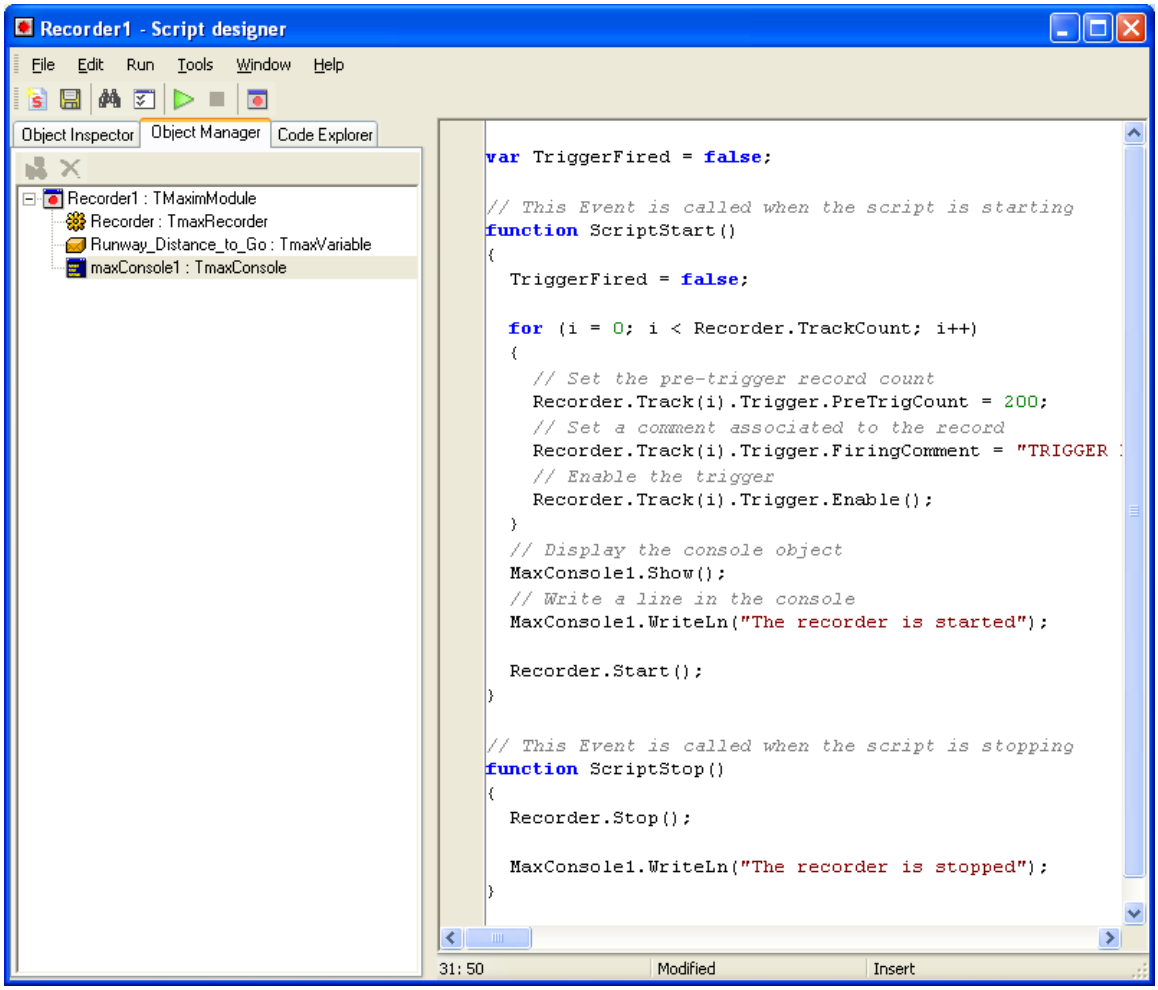
Let's write the ScriptStart event handler as follow:

```
// This Event is called when the script is starting
function ScriptStart()
{
  TriggerFired = false;
  for (i = 0; i < Recorder.TrackCount; i++)
  {
    // Set the pre-trigger record count
    Recorder.Track(i).Trigger.PreTrigCount = 200;
    // Set a comment associated to the record
    Recorder.Track(i).Trigger.FiringComment = "TRIGGER FIRED";
    // Enable the trigger
    Recorder.Track(i).Trigger.Enable();
  }
  // Display the console object
  MaxConsole1.Show();
  // write a line in the console
  MaxConsole1.WriteLn("The recorder started");

  Recorder.Start();
}
```

Write the ScriptStop event handler as follow:

```
// This Event is called when the recorder is stopping
function ScriptStop()
{
  MaxConsole1.WriteLn("The recorder stopped");

  Recorder.Stop();
}
```

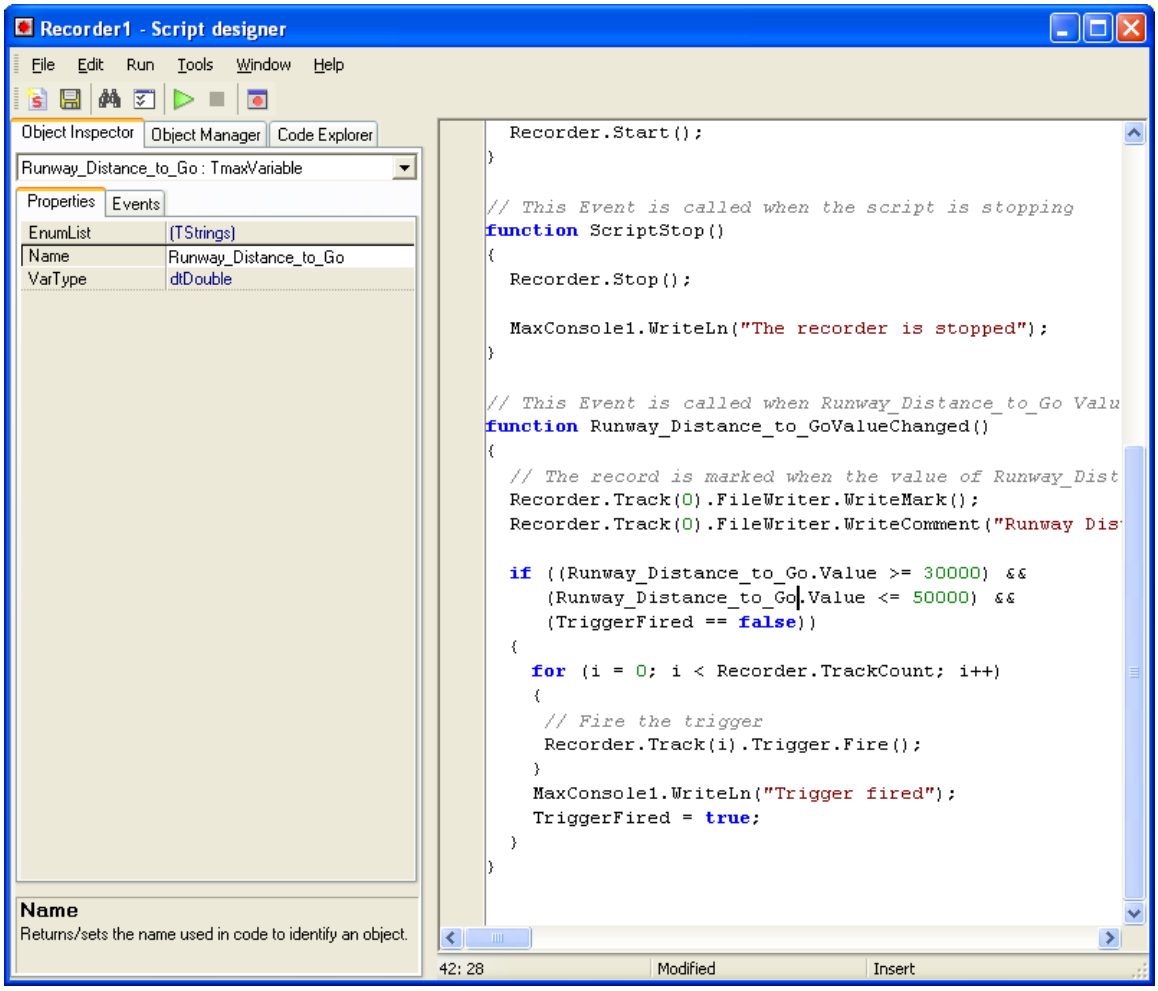
Note: a Recorder track corresponds to the channel defined in the recorder.



Select the "RunwayDistancetoGo" object to write the OnValueChanged event handler.

```
// This Event is called when RunwayDistanceToGo Value has changed
function RunwayDistanceToGoValueChanged()
{
  // The record is marked when the value of RunwayDistanceToGo has changed
  Recorder.Track(0).FileWriter.WriteMark();
  Recorder.Track(0).FileWriter.WriteComment("Runway Distance To Go value has changed");

  if ((RunwayDistanceToGo.Value >= 30000) &&
     (RunwayDistanceToGo.Value <= 50000) &&
     (TriggerFired == false))
```

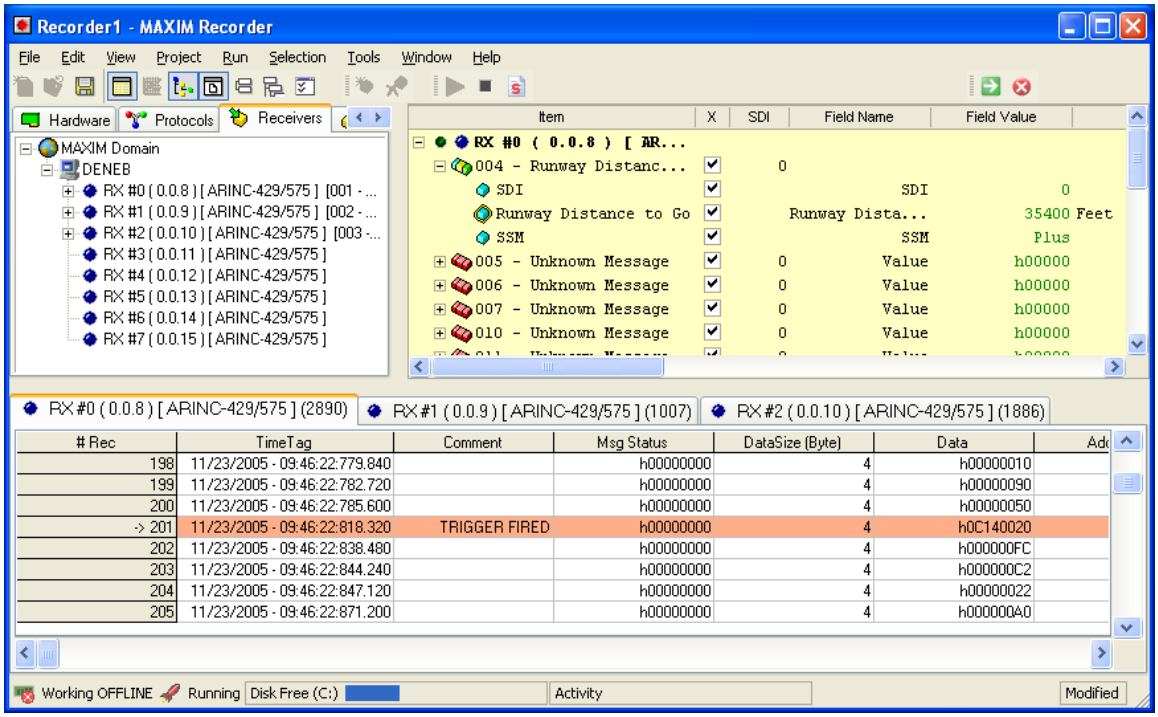
```
  {
    for (i = 0; i < Recorder.TrackCount; i++)
    {
     // Fire the trigger
     Recorder.Track(i).Trigger.Fire();
    }
    Console.WriteLn("Trigger fired");
    TriggerFired = true;
  }
}
```



Save the script.

Quit the script designing mode by choosing **Exit** from the **File** menu.

Click the Start button of the recorder. Once the "Runway Distance To Go" field value is within the trigger interval, the recorder session will be:



An equivalent script in VBScript would be:

```
Dim TriggerFired

'This Event is called when the script is starting
Sub ScriptStart

  TriggerFired = false
  For I = 0 to Recorder.TrackCount - 1
    'Set the pre-trigger record count
    Recorder.Track(i).Trigger.PreTrigCount = 1000
    'Set a comment associated to the record
    Recorder.Track(i).Trigger.FiringComment = "TRIGGER FIRED"
    'Enable the trigger
    Recorder.Track(i).Trigger.Enable
  Next
  'Display the console object
  MaxConsole1.Show
  'write a line in the console
  MaxConsole1.WriteLn("The recorder is started")
End Sub

'This Event is called when the script is stopping
Sub ScriptStop
  Console.WriteLn("The recorder is stopped")
End Sub

Sub RunwayDistanceToGoValueChanged
  'This Event is called when Runway_Distance_to_Go Value has changed

  ' The record is marked when the value of Runway_Distance_to_has changed
  Recorder.Track(0).FileWriter.WriteComment("Runway Distance To Go value has changed")
  Recorder.Track(0).FileWriter.WriteMark

  If (Runway_Distance_to_.Value > 30000) And (Runway_Distance_to_.Value < 50000) And
(TriggerFired = false) Then
    For I = 0 to Recorder.TrackCount - 1
     ' Fire the trigger
     Recorder.Track(i).Trigger.Fire
    Next
```

```
    MaxConsole1.WriteLn("Trigger fired")
    TriggerFired = true
  End If
End Sub
```

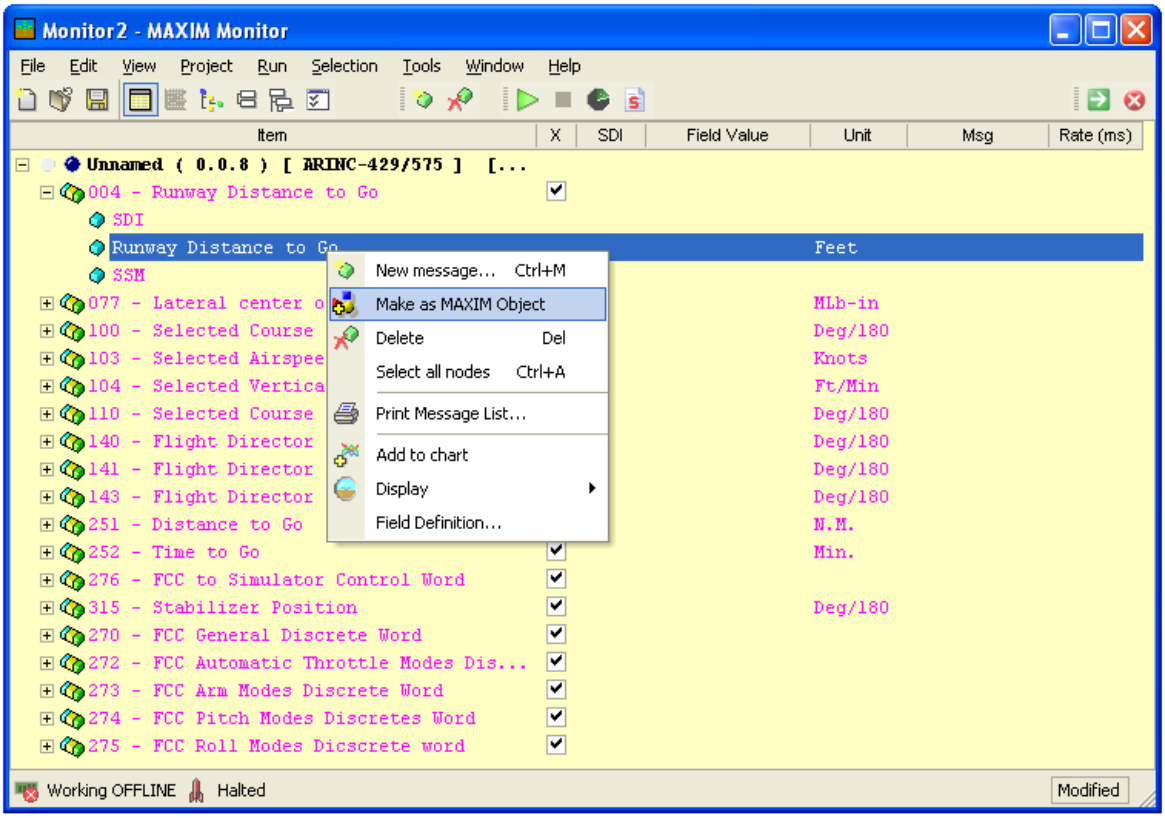
Appendix A
More Script explanation and examples

Scripts are introduced in MAXIM version 2.0 and let the user automate the control of MAXIM sessions and handle parameters received and transmitted in the context of a MAXIM Monitor, Scheduler and Recorder session.

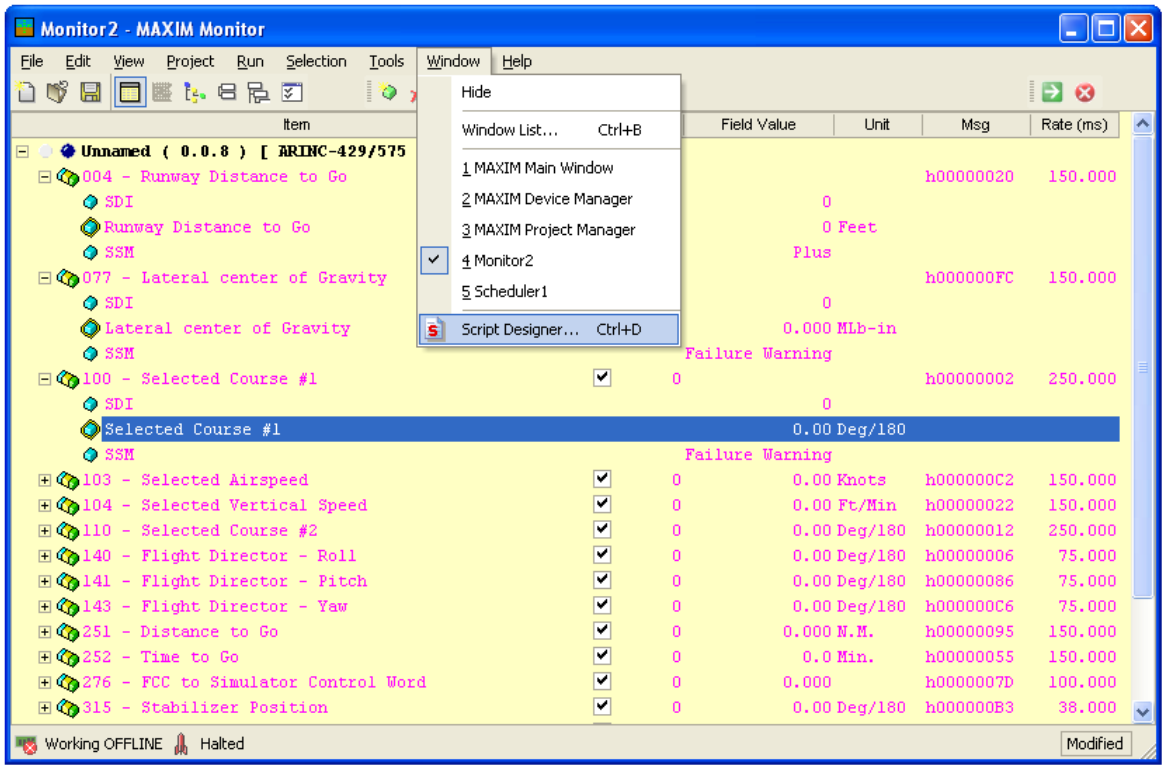
Construction of a MAXIM Script

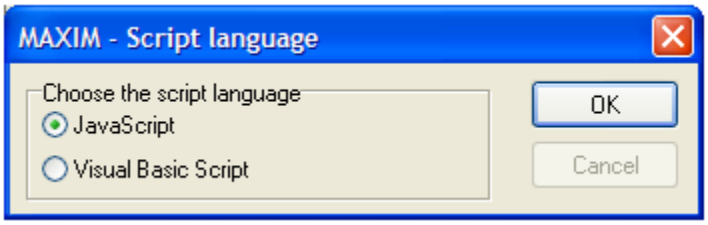
Before writing a Script in a MAXIM session, the user must initially choose objects (parameters) assigned to the MAXIM session to handle in the Script.

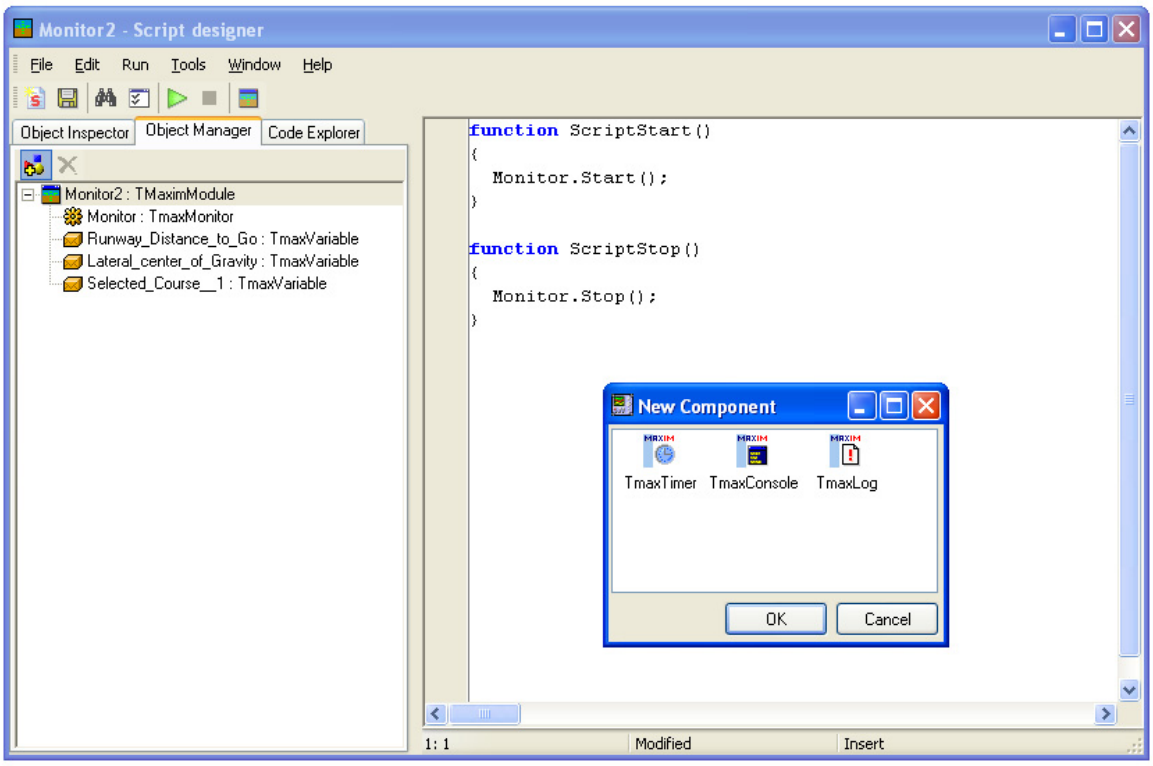
The selection is simply made by a right-click of the mouse on the selected element and select the **`Make as MAXIM Object`** command. The following figure shows how to select the parameter "Runway Distance to Go" and to expose it as a MAXIM object usable by a Script.



Once objects of the session are selected, open the Script Designer window using the **`Script Designer`** item in the **`Window`** menu as shown in the following figures:

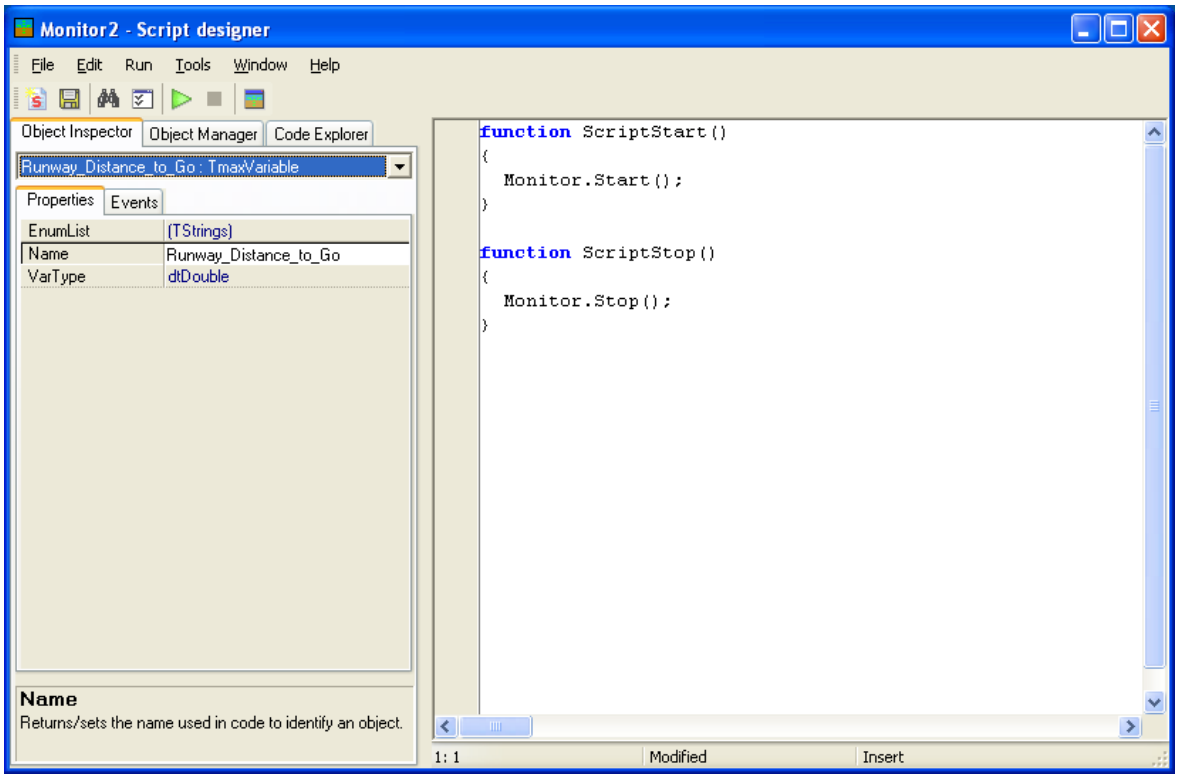

Monitor2 - MAXIM Monitor
File Edit View Project Run Selection Tools Window Help
Hide
Window List... Ctrl+B
1 MAXIM Main Window
2 MAXIM Device Manager
3 MAXIM Project Manager
4 Monitor2
5 Scheduler1
Script Designer... Ctrl+D
Item
Field Value
Unit
Msg
Rate (ms)
Unnamed (0.0.8) [ARINC-429/575
004 - Runway Distance to Go
SDI
Runway Distance to Go
SSM
077 - Lateral center of Gravity
SDI
Lateral center of Gravity
SSM
100 - Selected Course #1
SDI
Selected Course #1
SSM
103 - Selected Airspeed
104 - Selected Vertical Speed
110 - Selected Course #2
140 - Flight Director - Roll
141 - Flight Director - Pitch
143 - Flight Director - Yaw
251 - Distance to Go
252 - Time to Go
276 - FCC to Simulator Control Word
315 - Stabilizer Position
Working OFFLINE
Halted
Modified


MAXIM - Script language
Choose the script language
JavaScript
Visual Basic Script
OK
Cancel

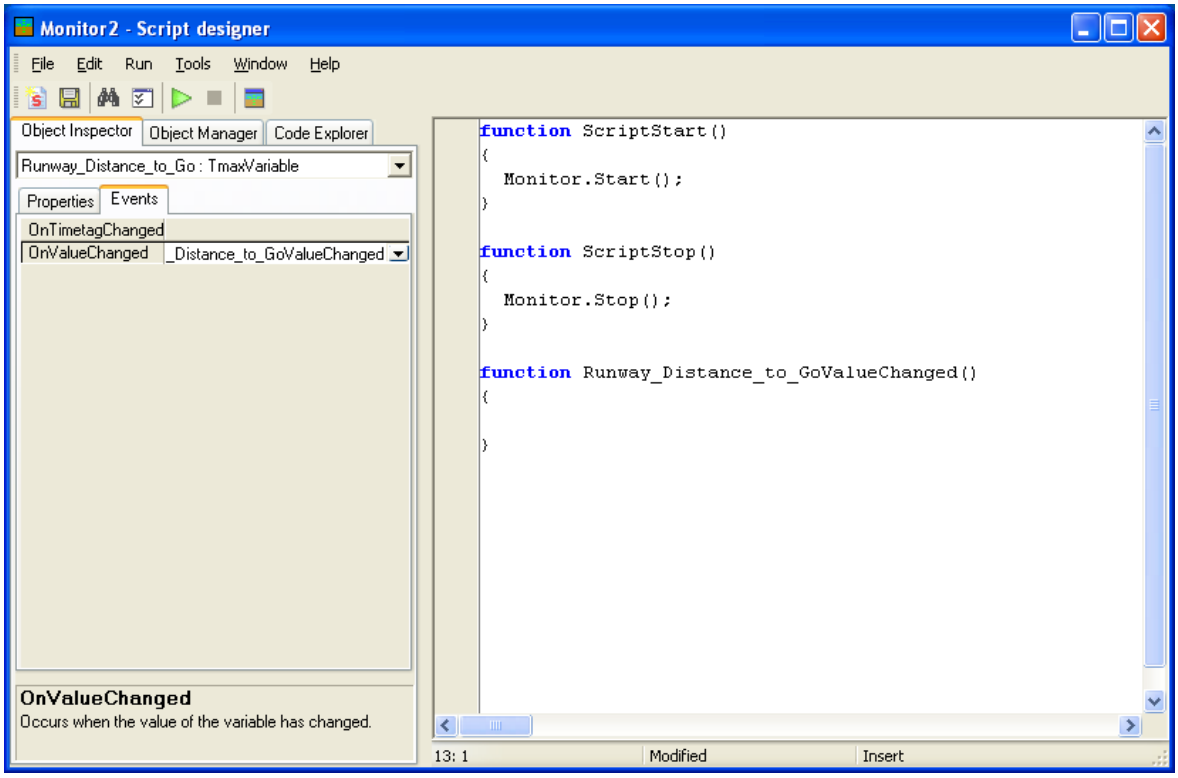


All objects exposed by MAXIM appear in the left section of the window, under the Object Manager tab. In this example, the Monitor Session object and three parameters are available. Use this tab to add or remove objects as shown using the New Component window. A TmaxConsole component will be added.

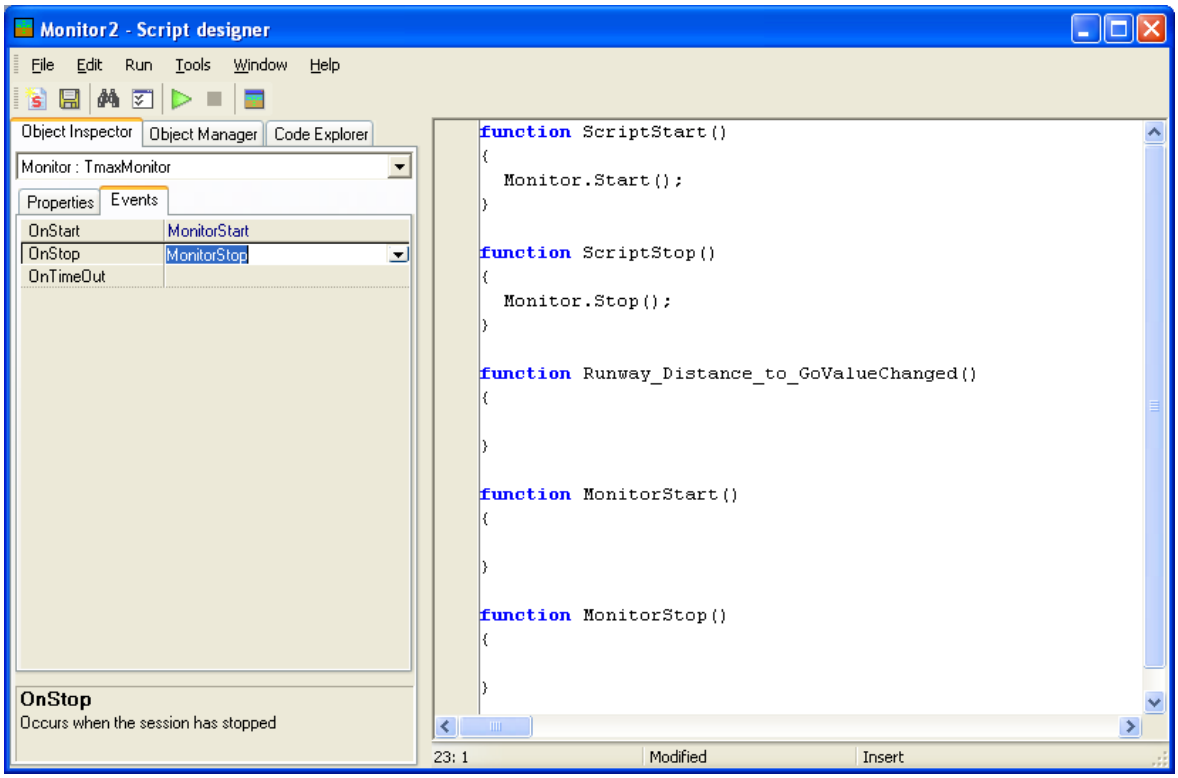
The Object Inspector tab list properties and events of the object selected in Object Manager tab as shown in the following figure:



To open an event handler signaled by an object, enter a function name to identify the event handler then click on it, or double-click in the empty zone so that a default name will be generated, and the skeleton of the function appears in the script editor. The following figure shows how to generate the event handler that will be called when the **`OnValueChanged`** event of the "Runway_Distance_to_Go" variable of type TmaxVariable occurs.

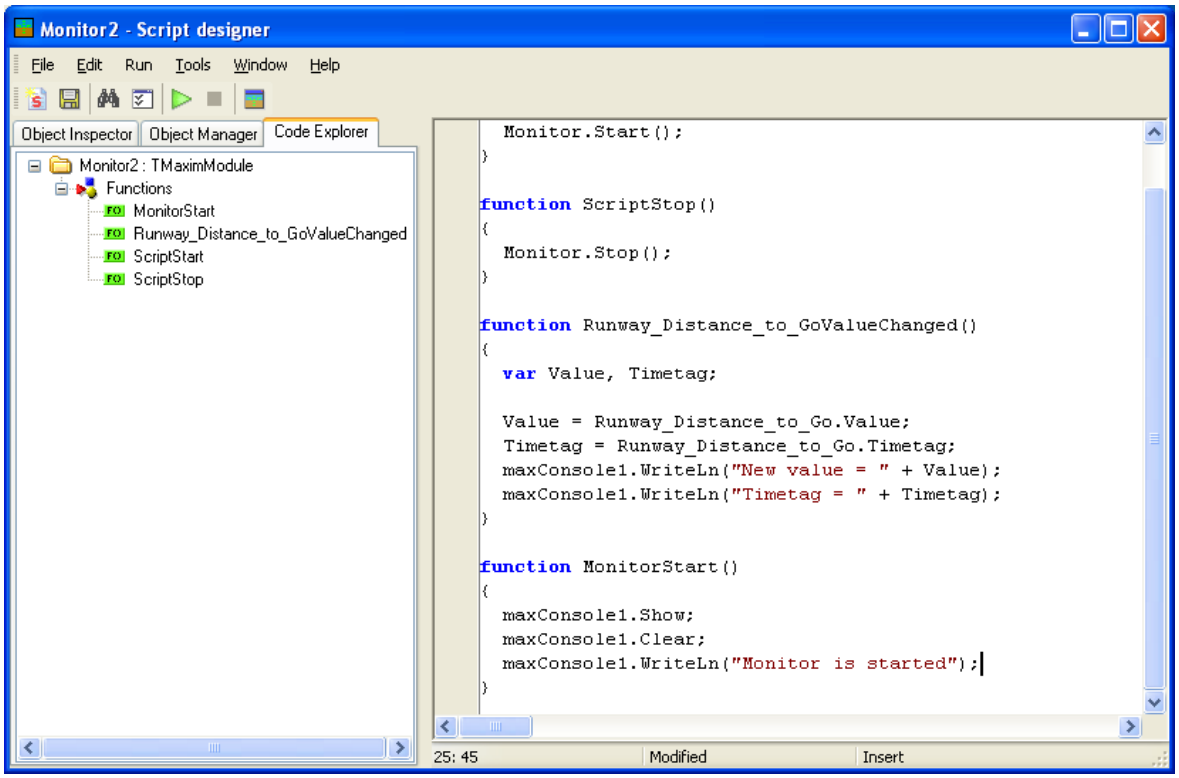


Next, the Session Monitor object is selected and functions to handle **`OnStart`** and **`OnStop`** events are added.

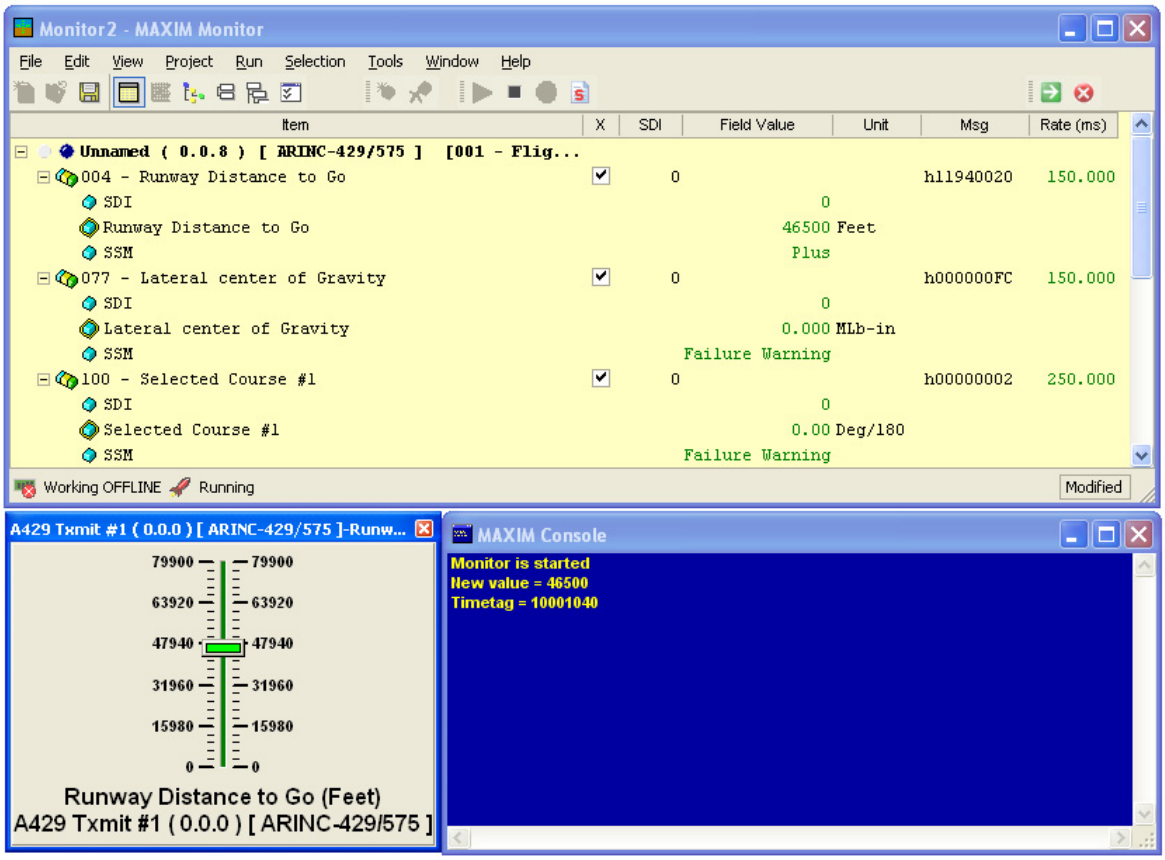


The script skeleton is now complete. It has the three newly added functions plus two automatic functions **`ScriptStart`** and **`ScriptStop`**. It is now time to give life to these functions by providing the code to perform the desired action.

For example, the WriteLn method of the Console object will be called in each function. It will post a message in the console in order to get feedback about event occurrences.



When the script programming for the Monitor session is done, launch the Monitor session. The Console window opens. It displays messages according to event handlers programmed in the script.



In MAXIM, each session (Monitor, Scheduler and Recorder) can have its own particular script. A script is written either in JavaScript (Jscript) or Visual Basic Script (VBScript) using the script editor window of a Script Designer session. The execution of a script allows:

- the use of exposed objects by MAXIM (session, parameters of a message), their methods, properties and events.
- the use of standard Windows objects (FileSystemObject object, dialog box, COM/DCOM, etc.) to have access to the PC's resources, to interact with the user, and to communicate to another Windows application.
- the use of a rich language syntax to create sequences, calculations, iterations, comparisons, decisions, etc…

MAXIM exposed objects

MAXIM exposes objects and their properties, methods and events. For example, these objects can be a communication bus parameter calculation, the signal of an analog converter, a recording session, a console window, a timer, etc...

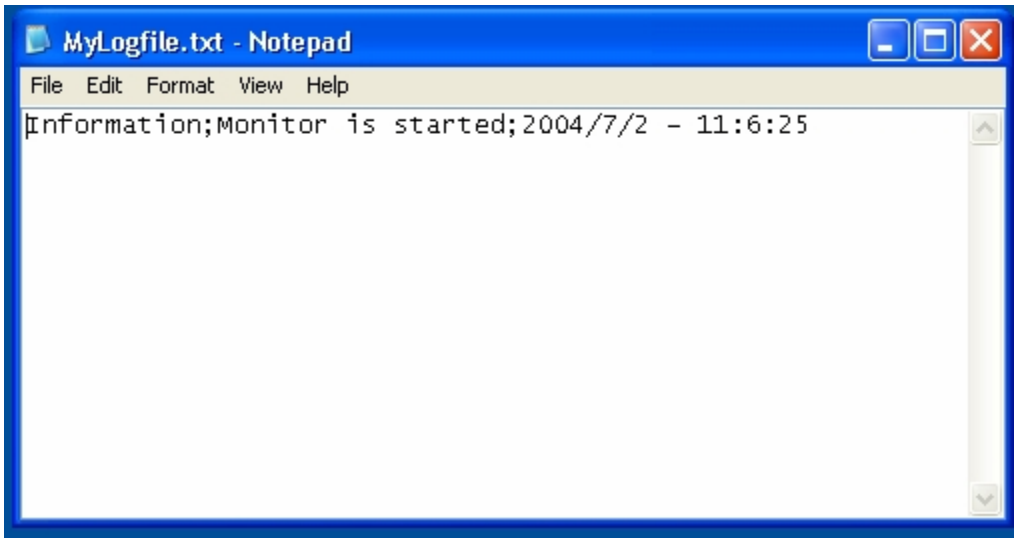
Windows objects

Several Windows objects can be accessed. It is possible to read or write in a file or to communicate with other Windows applications that support COM/DCOM services, and with Windows ActiveX objects.

The following function writes in a file:

```
//---------------------------------------------------------------
// FILE I/O
//---------------------------------------------------------------

function WriteToFile(filename, text)
{
   var ForReading = 1, ForWriting = 2; ForAppending = 8; var fso, file;

   dt = new Date();
   month = dt.getMonth()+1;
   day = dt.getDate();
   year = dt.getFullYear();
   hours = dt.getHours();
   minutes = dt.getMinutes();
   seconds = dt.getSeconds();

   // Access File System Object
   fso = new ActiveXObject("Scripting.FileSystemObject");

   // Open the file
   file = fso.OpenTextFile(filename, ForAppending, true);

   //Write text to the file
   file.Write("Information;");
   file.Write(text + ";");
   file.WriteLine( year + "/" + month + "/" + day + " - " +
               hours + ":" + minutes + ":" + seconds );
   file.Close();
}

function MonitorStart()
{
   var MyText = "Monitor is started";

   maxConsole1.Show;
   maxConsole1.Clear;
   maxConsole1.WriteLn("Monitor is started");
   WriteToFile("C:\\MyLogfile.txt", MyText);
}
```

Here is the text written in the MyLogFile.txt file when the Monitor session is launched:

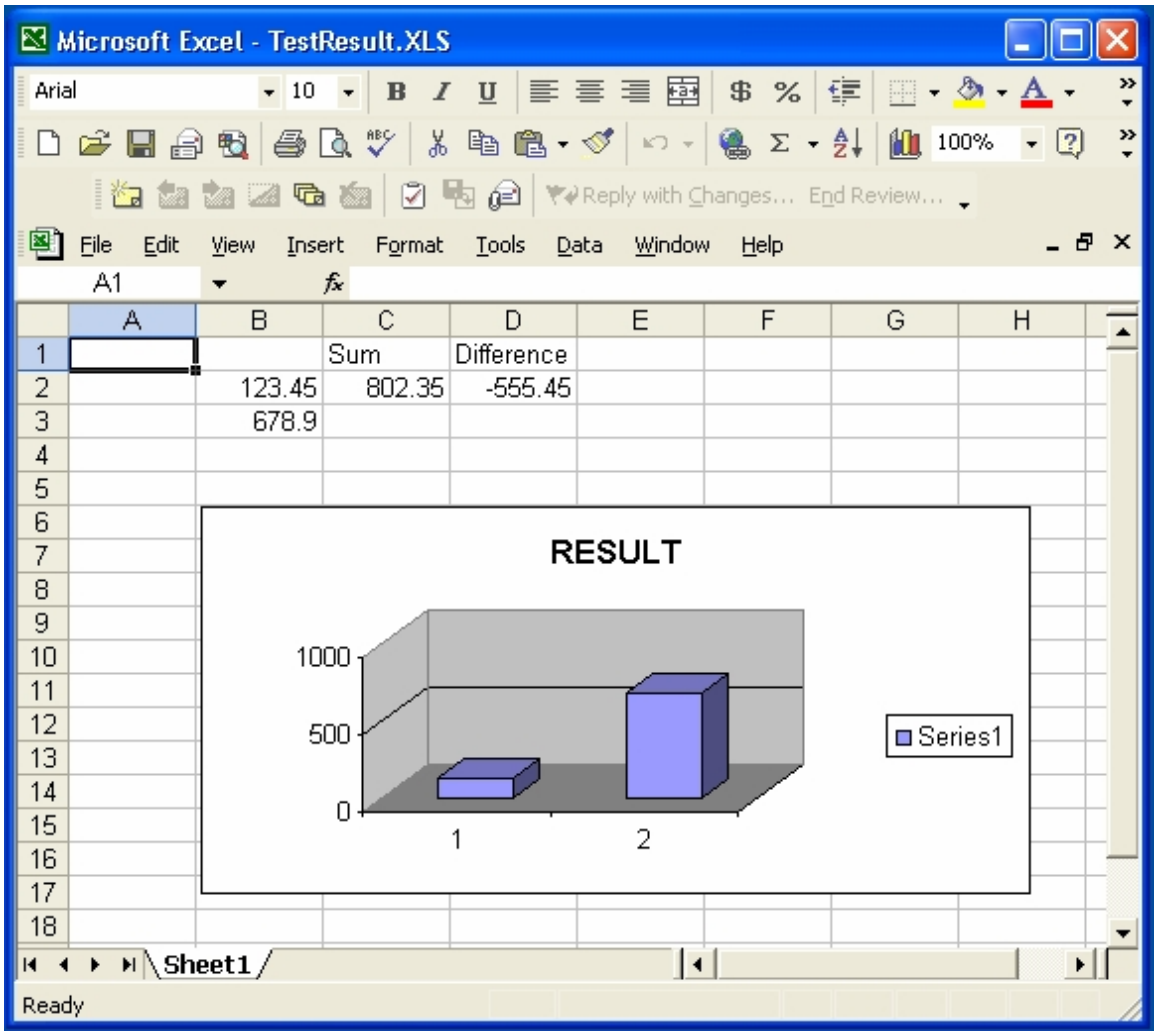


The following functions share data with MS-Excel when the Monitor session stops:

```
//---------------------------------------------------------------
// COMMUNICATION WITH MS-EXCEL
//---------------------------------------------------------------

function ExcelOpen()
{
   CellY=1;
   xlApp = new AtiveXObject("Excel.Application")
   xlApp.UserControl = true;
   xlBook = xlApp.Workbooks.Open("C:/TESTTEMPLATE.XLS");
   xlSheet = xlApp.ActiveSheet;
   xlSheet.Application.Visible = true;
}

function ExcelUpdate(Value)
{
   CellY = CellY +1;
   xlSheet.Application.Cells(CellY,2).Value = Value;
}

function ExcelClose()
{
   xlSheet.SaveAs("C:/TestResult.XLS");
   xlSheet.Application.Quit();
}

function MonitorStart()
{
   beep();
}
```

```
function MonitorStop()
{
   ExcelOpen();
   ExcelUpdate(123.45);
   ExcelUpdate(678.90);
   ExcelClose();
}
```

Here is the result when the Monitor session is stopped:



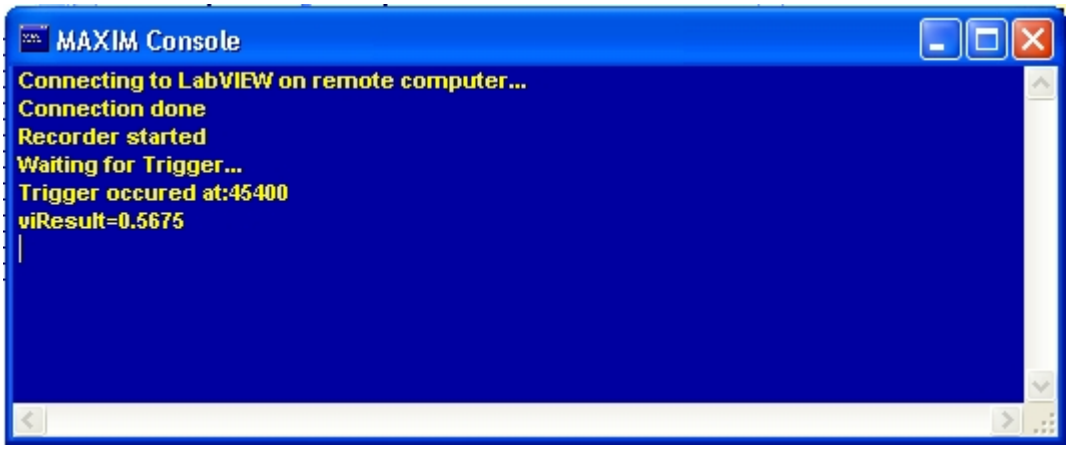
The following script controls a Recorder session and share data with labVIEW.

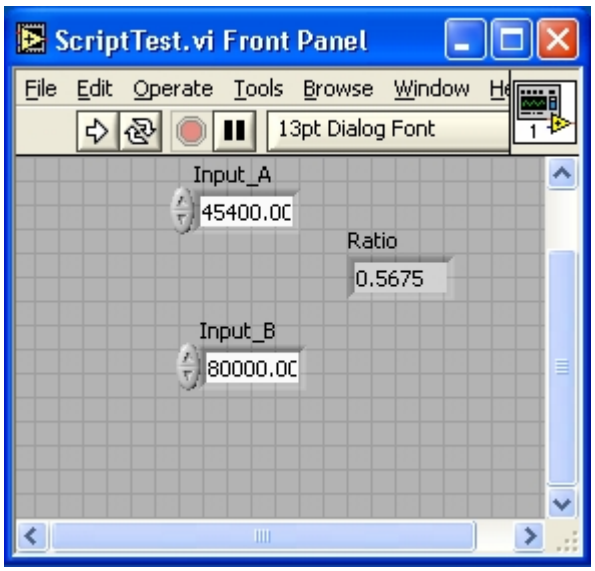
```
//---------------------------------------------------------------
// MAXIM Script
// SESSION : Recorder
// Language : JavaScript
//
// PURPOSE:
//
//    - SETUP RECORDER PRE-TRIG COUNT
//    - TRIG THE RECORDER WHEN ON MAXIM VARIABLE VALUE
//    - SEND DATA TO LABVIEW VI AND GET THE RESULT
//---------------------------------------------------------------


var trigger = false;
var lvapp;
var vi;
var connected;


//---------------------------------------------------------------
// LABVIEW REMOTE CONTROL
//---------------------------------------------------------------

function LabViewOpen()
{
   viPath = "C:ScriptTest.vi";
   try
   {
      lvapp = new ActiveXObject("LabVIEW.Application");
      vi = lvapp.GetVIReference(viPath); //Load the vi into memory
      vi.ShowFPOnLoad = true;
      vi.FPWinOpen = true;
      vi.FPAutoCenter = true;
      connected = true;
   }
   catch(e) // Exception occured when trying to establish connection with remote app.
   {
      connected = false;
   }
   return connected;
}

function LabViewUpdate(a,b)
{
   vi.SetControlValue("Input_A",a);
   vi.SetControlValue("Input_B",b);
   vi.Run(0);
   return vi.GetControlValue("Ratio");
}

function LabViewClose()
{
   lvapp.Quit();
   beep();
}
```

```
//---------------------------------------------------------------
// MAIN
//---------------------------------------------------------------

function RecorderStart()
{
   trigger = false;
   Console.Show();
   Console.Clear();
   Console.WriteLn("Connecting to LabVIEW on remote computer...");
   LabViewOpen();
   if (connected == true)
      Console.WriteLn("Connection done");
   else
      Console.WriteLn("Connection failed");

   // This Event is called when the recorder is starting
   for (i = 0; i < Recorder.TrackCount; i++)
   {
      // Set the pre-trigger record count
      Recorder.Track(i).Trigger.PreTrigCount = 200; // Set a comment associated to
                                                    // the record
      Recorder.Track(i).Trigger.FiringComment = "TRIGGER FIRED"; // Enable the
                                                                 // trigger
      Recorder.Track(i).Trigger.Enable();
   }
   // Display the console object

   // write a line in the console
   Console.WriteLn("Recorder started");
   Console.WriteLn("Waiting for Trigger...");
}

function RecorderStop()
{
   var viResult;
   LabViewClose();
   // This Event is called when the recorder is stopping
   Console.WriteLn("Recorder stopped");
   ShowMessage("Recorder Stopped - Press a key to continue");
   Console.Clear;
}

function RunwayDistanceToGoValueChanged()
{
   // This Event is called when RunwayDistanceToGo Value has changed
   var viResult;

   // The record is marked when the value of RunwayDistanceToGo has changed
   Recorder.Track(0).FileWriter.WriteMark();
   Recorder.Track(0).FileWriter.WriteComment("Runway Distance To Go value has changed");

   if ((RunwayDistanceToGo.Value >= 30000) &&
       (RunwayDistanceToGo.Value <= 50000) && (trigger == false))
   {
      trigger = true;
      for (i = 0; i < Recorder.TrackCount; i++)
      {
         // Fire the trigger
```

```
         Recorder.Track(i).Trigger.Fire();
      }
      Console.WriteLn("Trigger occured at:" + RunwayDistanceToGo.Value);

      if (connected == true)
      {
         viResult = LabViewUpdate(RunwayDistanceToGo.Value,80000);
         Console.WriteLn("viResult="+viResult);
      }
   }
}
```

Here is the result when the Monitor session is stopped:





Script languages and function libraries

Script languages supported by MAXIM have very rich sets of functions. Also, hundreds of libraries and examples of standard use of JScript and VBScript functions exist on the Web and can be re-used in the MAXIM environment.

The following example shows the use of mathematical functions to generate a variable signal like a sine wave or a ramp:

```
//-------------------------------------------------------------
// MATH EXEMPLE
//-------------------------------------------------------------

var PIX2 = 6.28318 // 3.14159 * 2.0
var TimeZero, First

function ComputeSinValue(Amplitude, Offset, Period, Timetag)
{
   if (First)
   {
      First = false;
      TimeZero = Timetag;
   }
   // Time Zero reference
   var Time = ((Timetag - TimeZero) / 1000000) // Time in msec
   Time %= 1000 * Period; // wrap time at zero when cycle completed
   Angle = (Time/(Period*1000)) * PIX2;
   return Offset + (Amplitude/2.0 * Math.sin(Angle));
}

function ComputeRampValue(Amplitude, Offset, Period, Timetag)
{
   if (First)
   {
      First = false;
      TimeZero = Timetag;
   }
   // Time Zero reference
   var Time = ((Timetag - TimeZero) / 1000000) // Time in msec
   Time %= 1000 * Period; // wrap time at zero when cycle completed
   return Offset + (Amplitude * Time/(Period*1000));
}


function RecorderStart()
{
   First = true;
   Console.Clear();
   Console.Show();
   Console2.Show();
   beep();
}

function RunwayDistanceToGoTimetagChanged()
{
   Value = ComputeSinValue(2000,0,10,RunwayDistanceToGo.Timetag);
   Console.WriteLn(Value);
   Value = ComputeRampValue(2000,0,10,RunwayDistanceToGo.Timetag);
```

```
   Console2.WriteLn(Value);
}

function RecorderStop()
{
   beep();
}
```

Conclusion

Preceding examples have shown the powerful capability of automation in MAXIM sessions using scripts and the use of objects both specific to MAXIM, and to the Windows operating system.

Scripts allow an easy, flexible and powerful way to control MAXIM sessions.